The University of Alberta

A Stable Algorithm for the Computation of Padé Approximants

by

Ronald John Meleshko

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Doctor of Philosophy

Department of Computing Science

Edmonton, Alberta
Spring 1990

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

ISBN 0-315-60388-7

NL-339 (r. 88/04) c

Canada

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Ronald John Meleshko
TITLE OF THESIS: A Stable Algorithm for the Computation of Padé Approximants
DEGREE: Doctor of Philosophy
YEAR THIS DEGREE GRANTED: 1990

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed) Ronald John Meleshko

Permanent Address:
57 Alderwood Boulevard
St. Albert, Alberta
Canada
T8N 3M4

Date: 18 April 1990

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **A Stable Algorithm for the Computation of Padé Approximants** submitted by **Ronald John Meleshko** in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

. . . . . .

(Supervisor)

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Date: . . . . . .

To

my parents
for the past,

Sharon
for the present,

and

Jana
for the future

Abstract

Padé approximants are rational function approximations to power series. The problem of obtaining a Padé approximant for a given power series is equivalent to solving a Hankel or Toeplitz system of equations. Many algorithms exist for obtaining Padé approximants using either the polynomial or the matrix formulation. Unfortunately, the numerical properties of these algorithms are such that even when the problem is well-conditioned, the algorithms can produce poor results unless additional restrictions are placed upon the problem being solved.

A new algorithm, NPADE, is presented that computes Padé approximants. It is shown that if the problem is well-conditioned, then NPADE produces a good solution; that is, the algorithm is shown to be weakly stable. No restrictions are imposed on the problem being solved. Except in certain pathological cases, the cost of the algorithm is $O(n^2)$, where n is the maximum degree of the polynomials comprising the Padé approximant.

The operation of the algorithm is controlled by a single parameter. This parameter is used in the expression of the bounds obtained for the error in the computed solution. Experimental results are given that show that the bounds, while crude, reflect the actual behavior of the error. In addition, it is shown how better bounds can easily be obtained *a posteriori*.

To show the usefulness of NPADE in applications, it is compared to a classical sequence accelerator, the ϵ-algorithm. It is demonstrated that NPADE can accelerate well-behaved sequences as accurately and with approximately the same cost as the ϵ-algorithm. For abnormal sequences, however, it is demonstrated that the ϵ-algorithm can fail, whereas NPADE continues to function just as effectively. As another application of NPADE, it is shown that it can be used to compute stably, in a weak sense, the inverse of a Hankel matrix.

Acknowledgements

I would like to express my sincere thanks to my supervisor, Stan Cabay. His insight into this area of numerical analysis was the source of many enjoyable and fruitful discussions over the years.

I would also like to thank the members of my examining committee, Dr. J. R. Bunch, Dr. H. J. Hoover, Dr. B. Joe, and Dr. S. D. Riemenschneider. Their comments helped to remove some of the rough edges in this thesis; I take full credit for the ones that remain.

John Kmet did a great job of proof-reading an earlier draft of the thesis. Special thanks go to the Departments of Computing Science and Mathematics for their support over the years and to Myrias Research Corporation for providing me time to write a good portion of the first draft of this thesis.

Finally, I must acknowledge the support of my family that I received throughout this endeavor, especially that of Sharon and my parents. Without their patience, kindness, and encouragement, this thesis would not have seen the light of day. Thank you!

Contents

1 Introduction **1**

1.1 A Brief Look at Padé Approximants 1

1.2 Overview of the Thesis 9

2 Padé Approximants **14**

2.1 Basic Definitions 14

2.2 Rissanen's Algorithm 21

2.3 The Algorithm of Cabay and Choi 28

3 Numerical Padé Approximant Algorithm **39**

3.1 Numerical Considerations 40

3.2 Stability 42

3.3 The Algorithm 48

4 Hankel Matrices and Their Inverses **53**

4.1 Inverses 54

5 Analysis of the Algorithm **63**

5.1 Preliminaries 65

5.2 Round-Off Analysis: The Individual Steps 69

5.2.1 Residual Calculation 72

5.2.2 Solution of the Sylvester System 73

5.2.3 Applying the Recurrence Relation 78

5.3 Cumulative Residual Error 80

5.4 Relative Errors and Stability 94

6 Experimental Results **102**

6.1 The Behavior of δR and δW 105

6.2 Errors in the Padé Approximants 108

7 Sequence Acceleration **120**

7.1 An Introduction to Sequence Acceleration 120

7.2 Experimental Results 125

8 Summary and Suggestions for Further Research **139**

A Details of Examples **145**

A.1 Tests of Chapter 6 145

A.2 Sequence Acceleration Test Cases 146

List of Tables

6.1 Test Case 1: $\varepsilon = 10^{-2}$ 109

6.2 Test Case 1: $\varepsilon = 10^{-12}$ 110

6.3 Test Case 2: $\varepsilon = 10^{-2}$ 111

6.4 Test Case 2: $\varepsilon = 10^{-12}$ 112

6.5 Test Case 3: $\varepsilon = 10^{-2}$ 113

6.6 Test Case 3: $\varepsilon = 10^{-12}$ 114

6.7 Test Case 1: $\| \delta U^{(k)} \| + \| \delta V^{(k)} \|$ 118

6.8 Test Case 2: $\| \delta U^{(k)} \| + \| \delta V^{(k)} \|$ 119

6.9 Test Case 3: $\| \delta U^{(k)} \| + \| \delta V^{(k)} \|$ 119

7.10 Test Case 4 131

7.11 Test Case 5 131

7.12 Test Case 6 132

7.13 Test Case 7 132

7.14 Test Case 8 133

7.15 Test Case 9 133

7.16 Test Case 10 134
7.17 Test Case 11 134
7.18 Test Case 12 135
7.19 Test Case 13 135
7.20 Test Case 14 135
7.21 Test Case 15 136
7.22 Test Case 16 136
7.23 Test Case 17 137
7.24 Test Case 18 137
7.25 Test Case 19 138

A.26 Series 1 149
A.27 Series 2 150
A.28 Series 3 151
A.29 Details of the Perturbation of Test Case 4 152

Chapter 1

Introduction

1.1 A Brief Look at Padé Approximants

Let

$$A(z) = \sum_{i=0}^{\infty} a_i z^i$$

be a formal power series where $a_i \epsilon \Re$ and $\Re$ denotes the real numbers. A problem in mathematics is that of approximating $A(z)$ by some other function or functions. One class of approximating functions is the class of rational functions over $\Re$. Once a class of approximating functions is selected, then the next question is how to define the approximation to the power series. One approach, dating back to the last century (Padé [46], Frobenius [23]), is to ensure that there is agreement between the coefficients of $A(z)$ and the

rational function. Loosely speaking[1], we say that $(U(z), V(z))$, where

$$U(z) = \sum_{i=0}^{m} u_i z^i \text{ and } V(z) = \sum_{i=0}^{n} v_i z^i,$$

is a Padé approximant of type (m, n) to $A(z)$ if

$$A(z)V(z) - U(z) = O(z^{m+n+1}). \tag{1.1}$$

Padé approximants find their use in many diverse fields of science. They can be used to model and approximate processes in control systems (Shamesh [50]), to study critical phenomena (Fisher [21]), and to analyse functions that have been measured experimentally (Pindor, [47]). They also play a role in numerical analysis. One major area of numerical analysis is the solution of partial differential equations. We consider the derivation of one technique in this area from the Padé approximant point of view, the Crank-Nicolson method (Baker and Graves-Morris [2]).

Example 1.1 A diffusion process can be mathematically modelled by the heat equation:

$$\frac{1}{\kappa}\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial z^2} = s(z,t)u. \tag{1.2}$$

The problem is to find a function $u(z,t)$ that satisfies (1.2) on some domain for which $t \geq 0$, subject to certain boundary conditions. The boundary conditions and the domain determine what method can be used to solve

[1] We defer formal definitions until the next chapter.

(1.2). If the z variable has a finite domain, $0 \leq z \leq L$, say, then (1.2) can be discretized over N equally spaced points, $(z_1, z_2, \ldots, z_N)$, in the z domain. Letting $u_i(t) = u(z_i, t)$, this yields

$$\frac{du_i}{dt} = \sum_{j=1}^{N} A_{ij} u_j(t) + S_i(t), \; i = 1, 2, \ldots, N, \tag{1.3}$$

where A_{ij} are constants resulting from the discretization and $S_i(t)$ are the discretized source terms (which also now depend upon the boundary conditions). To simplify the analysis, we shall assume that the source terms are zero.

To solve (1.3), the equation could now be discretized over time, but this places restrictions on the size of the time step, Δt, that may be unacceptable for certain problems[2]. Instead, a solution for (1.3) can be written as

$$U^{(k+1)} = \exp(A\Delta t) U^{(k)}, \tag{1.4}$$

where $U^{(k)} = (u_1(k\Delta t), \ldots, u_N(k\Delta t))^T$. Now the problem becomes one of how to evaluate the matrix exponential, $\exp(A\Delta t)$. One way is to evaluate the first l terms of the Maclaurin series for the matrix exponential. This method is unstable (Baker and Graves-Morris [2]), so another approach using the Maclaurin series would be to use the first l terms of $\exp(-A\Delta t)$ and then invert this matrix (i.e., compute $\{\exp(-A\Delta t)\}^{-1}$). Both of these approaches are computationally expensive (the latter more so, but it is more

[2]This arises when the problem is *stiff*.

stable). However, both expansions can be considered as Padé approximants; the former uses the $(l,0)$ Padé approximant for the exponential function and the latter the $(0,l)$ Padé approximant. The Padé approximants for the exponential function are well known (cf., Baker and Graves-Morris [3]), so this suggests the consideration of using other Padé approximants to evaluate the matrix exponential. In this case, the next simplest is the $(1,1)$ approximant which is:

$$\frac{1+\frac{1}{2}z}{1-\frac{1}{2}z}.$$

Using this in (1.4) gives us:

$$U^{(k+1)} = (1-\frac{1}{2}A\Delta t)^{-1}(1+\frac{1}{2}A\Delta t)U^{(k)} \tag{1.5}$$

which is an implicit method known as the Crank-Nicolson method. This method has certain useful numerical properties, such as being stable (Baker and Graves-Morris [2]).

While expression (1.5) can be derived from other considerations, this approach illustrates how Padé approximants can be used in numerical analysis; moreover, this use of Padé approximants can be generalized. To illustrate a generalization, we briefly consider another example from the area of partial differential equations.

Recently, Reusch et al [48] considered a model that describes the behavior of plasmas in controlled thermonuclear fusion experiments. The equation

they considered is:

$$\frac{\partial u}{\partial t} = H(t)u, \tag{1.6}$$

where $H(t)$ is a linear operator. This equation leads to the solution of

$$U(t,0) = \exp\left(\int_0^t dt H(t)\right). \tag{1.7}$$

In [48], they present a method for solving (1.7) that uses higher order diagonal Padé approximants for e^z. In order to perform the integration in (1.7), they use the approximants in a factored form; they report a significant gain in efficiency and accuracy over conventional methods. ∎

Padé approximants can be viewed as a matrix problem as well. From equation (1.1), we obtain the following system:

$$\begin{pmatrix} a_0 & & 0 & -1 & & 0 \\ & \ddots & & 0 & \ddots & \\ \vdots & & a_0 & & & -1 \\ & & \vdots & \vdots & & \vdots \\ a_{m+n} & & a_m & 0 & & 0 \end{pmatrix} \begin{pmatrix} v_0 \\ \vdots \\ v_n \\ u_0 \\ \vdots \\ u_m \end{pmatrix} = \vec{0}.$$

The system is under-determined and so has a nontrivial solution. We can

rewrite it as

$$\begin{pmatrix} 0 & \cdots & 0 & -1 & & 0 \\ a_0 & & & 0 & \ddots & \\ & \ddots & & & & -1 \\ \vdots & & a_0 & \vdots & & 0 \\ & & \vdots & & & \vdots \\ a_{m+n-1} & & a_m & 0 & & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \\ u_0 \\ \vdots \\ u_m \end{pmatrix} = -v_0 \begin{pmatrix} a_0 \\ \\ \vdots \\ \\ a_{m+n} \end{pmatrix}.$$

This system can be solved by decoupling $U(z)$ and $V(z)$. First a solution to

$$\begin{pmatrix} a_m & \cdots & a_{m-n+1} \\ \vdots & & \vdots \\ a_{m+n-1} & \cdots & a_m \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = -v_0 \begin{pmatrix} a_{m+1} \\ \vdots \\ a_{m+n} \end{pmatrix} \tag{1.8}$$

is obtained. This gives $V(z)$, which is then used in

$$\begin{pmatrix} u_0 \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} a_0 & & 0 \\ & \ddots & \\ \vdots & & a_0 \\ & & \vdots \\ a_m & \cdots & a_{m-n} \end{pmatrix} \begin{pmatrix} v_0 \\ \vdots \\ v_n \end{pmatrix} \tag{1.9}$$

to obtain $U(z)$. The matrix on the left-hand-side of equation (1.8) is called a Toeplitz matrix; it has the property that its entries along an arbitrary diagonal are identical. Now the order of the variables in (1.8) is arbitrary;

we could just as easily write it as:

$$\begin{pmatrix} a_{m-n+1} & \cdots & a_m \\ \vdots & & \vdots \\ a_m & \cdots & a_{m+n-1} \end{pmatrix} \begin{pmatrix} v_n \\ \vdots \\ v_1 \end{pmatrix} = -v_0 \begin{pmatrix} a_{m+1} \\ \vdots \\ a_{m+n} \end{pmatrix} \tag{1.10}$$

In this case, the matrix on the left-hand-side is called a Hankel matrix. We shall denote it by $H_{m,n}$.

It is clear that finding a solution to the Toeplitz or Hankel system defined by equations (1.8) or (1.10) is equivalent to solving the Padé approximant problem. It is this equivalence that allows us to view any problem involving Toeplitz or Hankel systems as an application of Padé approximants. To illustrate how Toeplitz systems arise in applications we consider an example from linear prediction (Makhoul [45], Cybenko [18]).

Example 1.2 In the linear prediction of discrete signals, one is given a doubly infinite sequence $(\ldots, s_{-1}, s_0, s_1, \ldots)$, where the subscript denotes the time of the discrete signal, and attempts to predict linearly s_i in terms of the n signals immediately preceeding it. The approximation to s_i takes the form:

$$\tilde{s}_i = -\sum_{j=1}^{n} v_{nj} s_{i-j}.$$

Using a least squares error measurement, the total error to be minimized is

$$E_n = \sum_{i=-\infty}^{\infty} (s_i + \sum_{j=1}^{n} v_{nj} s_{i-j})^2. \tag{1.11}$$

In the autocorrelation method, it is assumed that $0 < \sum_{i=-\infty}^{\infty} s_i^2 < \infty$ and E_n is minimized over $-\infty < i < \infty$. To minimize the expression in (1.11), partial derivatives with respect to the linear coefficients are taken and the resulting expressions are equated to zero. This yields:

$$\sum_{j=1}^{n} v_{nj} \sum_{i=-\infty}^{\infty} s_{i-j} s_{i-l} = -\sum_{i=-\infty}^{\infty} s_i s_{i-l}, \ 1 \le l \le n. \tag{1.12}$$

If we let $a_k = \sum_{i=-\infty}^{\infty} s_i s_{i-k}$, then equation (1.12) becomes:

$$\sum_{j=1}^{n} v_{nj} a_{l-j} = -a_l, \ 1 \le l \le n.$$

Observing that $a_l = a_{-l}$, we obtain in matrix form:

$$\begin{pmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_1 & & & \vdots \\ \vdots & & & a_1 \\ a_{n-1} & \cdots & a_1 & a_0 \end{pmatrix} \begin{pmatrix} v_{n1} \\ \vdots \\ \vdots \\ v_{nn} \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ \vdots \\ a_n \end{pmatrix}, \tag{1.13}$$

which is a symmetric Toeplitz system.

A symmetric Toeplitz system with a right-hand-side as in (1.13) is known as the Yule-Walker equation. This system arises in the analysis of sunspot data (Yule [67]) and in time series analysis (Walker [58]). Similar series arise in signal processing (Levinson [41]), linear prediction (Makhoul [45]), and linear filtering (Kailath [36]). ∎

1.2 Overview of the Thesis

This thesis is concerned with the computation of Padé approximants. Most of the algorithms described in the literature for computing Padé approximants are posed in an algebraic setting (Brent et al [6], Cabay and Choi [14], Bultheel [9]). There exist algebraic programming systems such as Maple and Mathematica that enable the easy coding of these algorithms. But algebraic algorithms are expensive and so they are limited, in practice, to small problems[3]. For large problems, which can arise in many areas (Jain [34]), the only recourse is to use numerical methods. In this case, the algebraic methods can be shown to be unstable for the general problem (Bultheel and Wuytack [10]). Some work has been done on the numerical aspects of their computation, but either it only addresses stability superficially and concentrates on numerical experiments (Luke [43], [44]) or else the analysis is only valid for restricted classes of problems (Wynn [65], Cybenko [18]). Bunch [11], [12] gives the best overview of the state of the art in the area; he provides an extensive survey of the limitations and shortcomings of many of the available methods. In this thesis we develop, present, and analyse an algorithm for computing Padé approximants that performs well numerically; moreover, it is robust in the sense that it is capable of recognizing problems as they arise. The results that we obtain are valid for the general class of

[3]See, for example, the timings reported in Geddes [24].

power series; no restrictions whatsoever are imposed.

To do this we need some background and understanding of the issues involved. This is done in Chapter 2. As we saw in Section 1.1, the Padé approximant problem is equivalent to solving a Toeplitz or Hankel system of equations. One would expect, then, that singularities in the corresponding systems would cause problems with the Padé approximants and their computations. This turns out to be the case; our presentation relies heavily on the corresponding matrix formulation. The degeneracies that can occur do have a regular structure; the algorithms that compute Padé approximants can be grouped into those that can and those that cannot deal with the degenerate cases. We present the ideas behind two algorithms in the algebraic setting that can handle the degenerate case and relate their performance to both the matrix and polynomial formulations of the problem.

The algorithms that can handle the degenerate cases do so by actually detecting them. Numerically, problems arise, however, if the problem is not degenerate, but nearly so. Chapter 3 starts with a discussion of how these problems can arise and relates them to algorithmic stability. Recognition of the near degenerate case is important for this stability (Graves-Morris [29]). A condition introduced in Chapter 2 that is equivalent to the nonsingularity of the corresponding matrix system will turn out to be an indicator of how close the problem is to degeneracy. At the end of Chapter 3, an algorithm is presented that makes use of this criteria to handle potentially unstable prob-

lems. What is presented is a stable algorithm for the computation of Padé approximants. To the best of our knowledge, it is the only such algorithm that takes advantage of the Padé structure to reduce the computational complexity of the problem and is stable in the general case. Other algorithms have been shown to be stable when the problem has an underlying positive-definite symmetric matrix (e.g. Cybenko [18]), but they are unstable for the general problem[4].

In Chapters 4 and 5, we present a numerical analysis of the algorithm. In the field there are only a limited number of error analyses for Padé and related algorithms (Wynn [65], Cybenko [18], Sweet [55]). Chapter 4 develops some of the theoretical material that is needed. A relationship is derived between intermediate results that are generated by the algorithm and the condition number of the corresponding matrix; this relationship forms the rationale for the stability parameter that is used by the algorithm. Also, although we require $U(z)$ and $V(z)$ such that

$$A(z)V(z) - U(z) = z^{m+n+1}W(z),$$

in practice we obtain instead $\tilde{U}(z)$ and $\tilde{V}(z)$ such that

$$A(z)\tilde{V}(z) - \tilde{U}(z) = \delta W(z) + z^{m+n+1}W(z)$$

[4]These algorithms solve the problem by solving subproblems. The matrices corresponding to these subproblems are submatrices of the original problem's associated matrix. These submatrices must also be well-conditioned for the algorithm to be stable. If the matrix is positive-definite symmetric and well-conditioned then its principal submatrices are well-conditioned as well. This is not true if the matrix is not positive-definite symmetric.

where $\partial(\delta W) = m+n$.[5] The analysis in Chapter 4 also suggests why δW can be a useful indicator of the "goodness" of $(\tilde{U}(z), \tilde{V}(z))$ as a Padé approximant to $A(z)$.

In Chapter 5, the actual steps that comprise the algorithm are analysed to give bounds on δW in terms of easily available quantities. This is then related to the errors in $U(z)$ and $V(z)$, enabling us to make assertions about the stability of the algorithm. We see that the errors are directly related to the stability parameter we alluded to earlier. The algorithm uses the size of this parameter as a criteria for deciding whether or not a computed solution is good. If it is not good, then the underlying subproblem was unstable and the solution is rejected. The algorithm then solves a slightly more complex system in an attempt to find a subproblem that is stable. In a worst case scenario, the algorithm is equivalent to Gaussian elimination, but in most cases, it is a fast $O(n^2)$ algorithm.

The bounds presented in Chapter 5 have large constants in them. In Chapter 6, we present experimental results that show that the bounds are qualitatively correct, but the size of the constants causes overestimation of the errors. A comment by J. H. Wilkinson [59] puts all this in perspective. The essence of what he says is that the main purpose of an error bound is usually to help one understand an algorithm and to see where it can be im-

[5] We use $\partial(P(z))$ to denote the degree of the polynomial $P(z)$.

proved; the bound is usually not of great use due to the severe overestimates that occur during the analysis.

In Chapter 6, we take this to heart and alter the algorithm's criteria slightly and apply it in one area of Padé approximation - sequence acceleration. We present some background on sequence acceleration and then report on some of the results of the algorithm applied to sequences from the literature and compare them with results generated by another (well studied) algorithm, namely the ϵ-algorithm. The advantage of making comparisons to the ϵ-algorithm is that it has been extensively compared to other sequence accelerators and has been shown to be be a most useful accelerator for many sequences (Smith and Ford [53]). We see that for a large number of experiments our algorithm always performs as well as the ϵ-algorithm and has the advantage of being able to handle the near degenerate cases. Unfortunately, there is limited work in the literature dealing with the ϵ-algorithm and degenerate cases (see, for example, Wynn [64]). Thus, we not only have an algorithm for Padé approximants that is stable, but a sequence accelerator that will perform well in the presence of degeneracies.

Chapter 2

Padé Approximants

In this chapter, we give formal definitions and present some results that will be useful later on; all these results are from the literature. We also survey some of the Padé approximation algorithms and relate them to Gaussian elimination.

2.1 Basic Definitions

Let

$$A(z) = \sum_{i=0}^{\infty} a_i z^i \text{ and } B(z) = \sum_{i=0}^{\infty} b_i z^i,$$

be formal power series, where $a_i, b_i \epsilon \Re$. We assume, without loss of generality, that $b_0 \neq 0$. (If $b_0 = 0$ and $a_0 \neq 0$, then $A(z)$ and $B(z)$ can be interchanged. If both are zero, then if b_λ (or a_λ) is the first nonzero coefficient, we can work

with $z^{-\lambda}A(z)$ and $z^{-\lambda}B(z)$.) We are interested in a certain type of rational approximant to the power series quotient $A(z)/B(z)$ given by:

Definition 2.1 *Let*

$$U(z) = \sum_{i=0}^{m} u_i z^i \text{ and } V(z) = \sum_{i=0}^{n} v_i z^i$$

be polynomials of degrees m and n, respectively. $(U(z), V(z))$ is said to be a **Padé form of type** (m,n) *for the power series quotient $A(z)/B(z)$ if $V(z)$ is nonzero and*

$$A(z)V(z) + B(z)U(z) = z^{m+n+1}W(z). \tag{2.1}$$

$W(z)$ is called the **residual** *for the (m,n) Padé form.*

This definition is more general than that presented informally in Chapter 1. In practice, the case where $B(z) \equiv -1$ is most often considered. We make use of this more general formulation in the presentation of some of the later material.

The Sylvester matrix is defined to be:

$$\tilde{S}_{m,n} = \begin{pmatrix} a_0 & & 0 & b_0 & & 0 \\ & \ddots & & & \ddots & \\ \vdots & & a_0 & \vdots & & b_0 \\ \vdots & & \vdots & \vdots & & \vdots \\ & & \vdots & & & \vdots \\ a_{m+n} & \cdots & a_m & b_{m+n} & \cdots & b_n \end{pmatrix}.$$

Then (2.1) is equivalent to

$$\bar{S}_{m,n} \begin{pmatrix} v_0 \\ \vdots \\ v_n \\ u_0 \\ \vdots \\ u_m \end{pmatrix} = 0. \tag{2.2}$$

Theorem 2.2 *For any power series quotient $A(z)/B(z)$, there exists a Padé form, $(U(z), V(z))$, of type (m,n).*

Proof Since $\bar{S}_{m,n}$ is a $(m+n+1)$ by $(m+n+2)$ matrix, (2.2) is underdetermined and has a nontrivial solution. If $V(z) \equiv 0$, then it is easily seen that we must have $U(z) \equiv 0$ as well, since $b_0 \neq 0$. But this contradicts the existence of a nontrivial solution; therefore, $V(z)$ must be nonzero. ∎

A useful way to characterize the collection of the Padé forms for a given A and B is:

Definition 2.3 *The* **Padé table** *for $A(z)/B(z)$ is the doubly infinite array whose (m,n) entry is a Padé form of type (m,n) for A/B.*

(For descriptive purposes, it is the common practice (Gragg [28]) to consider the $(0,0)$ entry in the table to be in the upper left corner, so that the table

extends infinitely downwards and to the right.) With this definition, we see that the Padé table entries need not be unique. If we let

$$S_{m,n} = \begin{pmatrix} 0 & & 0 & b_0 & & 0 \\ a_0 & & & & \ddots & \\ \vdots & \ddots & & \vdots & & b_0 \\ \vdots & & a_0 & \vdots & & \vdots \\ & & \vdots & & & \vdots \\ a_{m+n-1} & \cdots & a_m & b_{m+n} & \cdots & b_n \end{pmatrix}$$

($S_{m,n}$ is just $\tilde{S}_{m,n}$ with its first column deleted), then a solution to (2.2) can be obtained from

$$S_{m,n} \begin{pmatrix} v_1 \\ \vdots \\ v_n \\ u_0 \\ \vdots \\ u_m \end{pmatrix} = -v_0 \begin{pmatrix} a_0 \\ a_1 \\ \\ \vdots \\ \\ a_{m+n} \end{pmatrix}. \tag{2.3}$$

If $S_{m,n}$ is nonsingular, then it is clear from equation (2.3) that the Padé form is unique up to multiplication of U and V by a scalar and that $v_0 \neq 0$. The converse is not true; the (m,n) Padé form can be unique up to multiplication by a scalar, but $S_{m,n}$ can still be singular (in this case, however, v_0 will be zero). Moreover, if $S_{m,n}$ is singular and the solution is not unique, then v_0 may or may not be nonzero.

To ensure uniqueness of a given Padé form conditions are sometimes imposed. However, care must be taken, as the conditions may preclude the existence of the Padé form for some (m,n). For example, we could require that the Padé form be such that $U(z)$ and $V(z)$ are relatively prime with $V(0) = 1$. This condition gives us what is termed a Padé fraction.

Definition 2.4 *A power series is said to be* **normal** *if $S_{m,n}$ is nonsingular for all $m, n \geq 0$.*

If the series is normal, then there will be a unique Padé fraction for each entry in the Padé table. However, if the power series is not normal, then the Padé fraction may no longer exist for some (m,n).

Normality for a power series and, hence, uniqueness of its Padé table entries is a useful property, but even if the series is not normal, the Padé table still has a regular structure as the following lemma shows.

Lemma 2.5 *Let $(U(z), V(z))$ be an (m,n) Padé form for $A(z)/B(z)$ such that*

$$A(z)V(z) + B(z)U(z) = z^{m+n+s}W(z),$$

where $W(0) \neq 0$, $s \geq 1$, $\partial(U) = m$, and $\partial(V) = n$. If α, β are nonnegative integers such that $\alpha + 2\beta \leq s - 1$, then every $(m+\alpha+\beta, n+\alpha+\beta)$ Padé form can be written as

$$(z^{\alpha}G(z)U(z), z^{\alpha}G(z)V(z)),$$

where $G(z)$ is an arbitrary polynomial such that $G(0) \neq 0$ and $\partial(G) = \beta$.

Proof

We will just show that $(z^\alpha G(z)U(z), z^\alpha G(z)V(z))$ is a $(m+\alpha+\beta, n+\alpha+\beta)$ Padé form for A/B. A proof of the converse can be found in Gragg [28, Theorem 3.2, page 13].

Clearly, $\partial(z^\alpha G(z)U(z)) = (m+\alpha+\beta)$, $\partial(z^\alpha G(z)V(z)) = (n+\alpha+\beta)$, and $(z^\alpha G(z)V(z)) \neq 0$. Now

$$\begin{aligned} A(z)(z^\alpha G(z)V(z)) + B(z)(z^\alpha G(z)U(z)) &= z^\alpha G(AV+BU) \\ &= z^\alpha G(z^{m+n+s}W(z)) \\ &= z^{(m+\alpha+\beta)+(n+\alpha+\beta)+s-\alpha-2\beta}WG \\ &= O(z^{(m+\alpha+\beta)+(n+\alpha+\beta)+1}), \end{aligned}$$

so that $(z^\alpha G(z)U(z), z^\alpha G(z)V(z))$ is a $(m+\alpha+\beta, n+\alpha+\beta)$ Padé form for A/B. ∎

In Gragg [28], it is actually shown that the singular $S_{m,n}$ occur in square blocks within the Padé table and that all solutions in these singular blocks are just certain multiples of the Padé form that occurs in the upper-left corner of the block. Thus all the Padé forms occurring at an entry where the solution is nonunique can be obtained by finding the unique Padé form for the start of its block. Also, once s is determined all solutions in that block can be found.

Another observation is that at the end of these blocks, the entry is unique as well. This is summed up by the following lemma.

Lemma 2.6 *If $S_{m,n}$ is nonsingular, then the $(m-1,n-1)$ Padé form is unique up to multiplication by a scalar constant.*

Proof Consider the equation

$$\begin{pmatrix} a_0 & & 0 & b_0 & & 0 \\ & \ddots & & & \ddots & \\ \vdots & & a_0 & \vdots & & \\ & & \vdots & & & b_0 \\ & & \vdots & \vdots & & \vdots \\ a_{m+n-1} & & a_m & b_{m+n-1} & & b_n \end{pmatrix} \begin{pmatrix} q_0 \\ \vdots \\ q_{n-1} \\ p_0 \\ \vdots \\ p_{m-1} \end{pmatrix} = \begin{pmatrix} 0 \\ \\ \vdots \\ \\ 0 \\ r_0 \end{pmatrix}. \qquad (2.4)$$

A solution to (2.4) then satisfies the equation

$$A(z)Q(z) + B(z)P(z) = z^{m+n-1}(r_0 + r_1 z + \ldots) = z^{m+n-1}R(z) \qquad (2.5)$$

which is one of the conditions defining the $(m-1,n-1)$ Padé form; the others follow trivially. But the matrix in equation (2.4), call it S^-, is just $S_{m,n}$ with its $(n+1)^{st}$ column and first row deleted. Therefore,

$$\det(S_{m,n}) = (-1)^n b_0 \det(S^-),$$

so S^- must be nonsingular as $b_0 \neq 0$. Thus, if r_0 is a fixed nonzero value, then the solution of (2.4) gives a unique $(m-1,n-1)$ Padé form. ∎

Theorem 2.7 *Let (U,V) be a (m,n) Padé form for A/B satisfying (2.1) and let (P,Q) be a $(m-1,n-1)$ Padé form for A/B satisfying (2.5). Then*

$$r_0 v_0 \neq 0 \Leftrightarrow S_{m,n} \text{ is nonsingular.}$$

Proof The fact that $S_{m,n}$ being nonsingular implies that $r_0 v_0 \neq 0$ follows from Lemma 2.6 and the comments concerning equation (2.3). For a proof of the converse, see Labahn, Choi, and Cabay [39]. ∎

Theorem 2.7 shows the importance of r_0 and v_0 in determining whether or not $S_{m,n}$ is nonsingular. We will see in Chapter 4 that they can also be used to determine the condition number of $S_{m,n}$. This relationship will prove to be critical in describing the properties of the numerical algorithm that we introduce in Chapter 3.

2.2 Rissanen's Algorithm

Traditionally, Padé approximants have dealt with the case $B(z) \equiv -1$; we used a more general formulation in the previous section where the only restriction on $B(z)$ was that $b_0 \neq 0$. But the former is equivalent to the latter as we can formally divide $A(z)$ by $-B(z)$. So for the moment, we return to the traditional view and assume that $B(z) \equiv -1$. In this case, as we saw in Chapter 1, the problem of obtaining the (m,n) Padé approximant is equivalent to solving a Hankel or Toeplitz system of equations. We stress this

equivalence as the literature we cite in the remainder of this chapter deals with Toeplitz and Hankel systems as well as Padé approximants.

As the Padé approximant problem can be stated as a problem in matrix algebra, solutions can be obtained using matrix methods. For example, Gaussian elimination can be used; in this case, we have an $O(n^3)$ algorithm. However, the linear system being solved here has a special structure and this can be exploited to obtain algorithms of lower order cost. We now consider one such method.

While the approach we present is matrix-oriented, we will emphasize the connection with the polynomial presentation of the previous section. Since $B(z) \equiv -1$, the major computational effort in obtaining the (m,n) Padé approximant involves solving the Hankel system

$$\begin{pmatrix} a_{m-n+1} & \cdots & a_m \\ \vdots & & \vdots \\ a_m & \cdots & a_{m+n-1} \end{pmatrix} \begin{pmatrix} v_n \\ \vdots \\ v_1 \end{pmatrix} = -v_0 \begin{pmatrix} a_{m+1} \\ \vdots \\ a_{m+n} \end{pmatrix} \tag{2.6}$$

Rissanen's algorithm [49] takes the Hankel matrix, $H_{m,n}$, and calculates a lower triangular matrix P such that that $PH_{m,n} = G$, where G is a row permuted upper triangular matrix. It does this iteratively by successively producing such decompositions for certain submatrices of $H_{m,n}$. In particular, it does this for successive nonsingular principal submatrices of $H_{m,n}$. It exploits the Hankel structure to reduce the number of operations performed; to do this, however, it uses Gaussian elimination without pivoting.

The order i principal submatrix of $H_{m,n}$ is given by:

$$\begin{pmatrix} a_{m-n+1} & \cdots & a_{m-n+i} \\ \vdots & & \vdots \\ a_{m-n+i} & \cdots & a_{m-n+2i-1} \end{pmatrix}, \tag{2.7}$$

which, we see upon examination, is the Hankel matrix, $H_{m-n+i,i}$, associated with the $(m-n+i,i)$ Padé form.[1] This observation allows us to show that the decomposition of successive principal submatrices is equivalent to obtaining successive Padé forms along the $(m-n)$ diagonal of the Padé table. To see this, suppose that (2.7) is nonsingular. Then there exist (P,Q) and (U,V) which are the $(m-n+i-1,i-1)$ and $(m-n+i,i)$ Padé forms for $A(z)$, respectively, with $r_0 \neq 0$ and $v_0 \neq 0$. (This follows from Theorem 2.7 and the fact that $H_{m,n}$ is nonsingular if and only if $S_{m,n}$ is nonsingular.) In matrix notation, we can then write:

$$\begin{pmatrix} q_{i-1} & \cdots & \cdots & q_0 & 0 \\ v_i & \cdots & \cdots & v_1 & v_0 \end{pmatrix} \begin{pmatrix} a_{m-n+1} & \cdots & a_m \\ \vdots & & \vdots \\ a_{m-n+i} & \cdots & a_{m+i-1} \\ a_{m-n+i+1} & \cdots & a_{m+i} \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \cdots & r_0 & & \cdots & r_{n-i+1} \\ 0 & \cdots & 0 & w_0 & \cdots & w_{n-i} \end{pmatrix}, \tag{2.8}$$

[1] If $m < n$, then we assume, without loss of generality, that $i \geq n-m$. When $i = n-m$, the Hankel matrix is, trivially, a row permuted upper triangular matrix and the decomposition is immediate.

where we have extended the $(i+1)^{st}$ principal submatrix of $H_{m,n}$ to be an $(i+1)$ by n matrix consisting of the first $(i+1)$ rows of $H_{m,n}$; we shall denote this submatrix by $H(i+1)$. If all smaller submatrices $H_{m-n+j,j}$, $\max\{0, n-m\} \leq j < i$, are nonsingular, then the first matrix on the left hand side of (2.8) can be built up to be an $(i+1)$ by $(i+1)$ matrix where the j^{th} row is the denominator of the $(m-n+j-1, j-1)$ Padé form. Then the corresponding $(i+1)$ by n matrix on the right hand side consists of the residuals that result from each Padé form. The former will be lower triangular and the latter will be upper trapezoidal. This is what Rissanen's algorithm produces at an intermediate stage of operation. We illustrate how the algorithm proceeds to the next stage and, in so doing, show how a singular submatrix is handled. To continue, we only need the portion of the decomposition given in (2.8).

The leading entries of $W(z)$ are examined to find the first entry that is nonzero. Suppose it is w_{s-1}. Then, to perform the next step, the first matrix on the left hand side is increased to an $(i+s+1)$ by $(i+s+1)$ matrix and s rows are added to the other two matrices , so that we have:

$$\begin{pmatrix} q_{i-1} & \cdots & q_0 & & \\ v_i & \cdots & v_1 & v_0 & \\ & \ddots & & & \ddots \\ & & v_{\tilde{r}} & \cdots & v_0 \end{pmatrix} H(i+s+1) \qquad (2.9)$$

$$= \begin{pmatrix} 0 & \cdots & r_0 & r_1 & \cdots & & & r_{n-i+1} \\ & & & & 0 & w_{s-1} & \cdots & w_{n-i} \\ & & & & w_{s-1} & & & w_{n-i+1} \\ & & & \iddots & & & & \vdots \\ 0 & \cdots & w_{s-1} & \cdots & & w_{2s-1} & \cdots & w_{n-i+s} \end{pmatrix}$$

We shall denote the first matrix on the left-hand-side by $L(i+s+1)$ and the matrix on the right-hand-side by $U(i+s+1)$. Due to the structure of the Hankel system only s new entries need to be calculated as all other entries in the new rows of $U(i+s+1)$ can be obtained by shifting the entries in previous rows. Since $r_0 \neq 0$ and $w_{s-1} \neq 0$, row operations can then be performed to obtain:

$$\begin{pmatrix} q_{i-1} & \cdots & q_0 & & & \\ v_i & \cdots & v_1 & v_0 & & \\ & \ddots & & & \ddots & \\ & & v_i & \cdots & & v_0 \\ * & * & & \cdots & * & * \end{pmatrix} H(i+s+1) \tag{2.10}$$

$$= \begin{pmatrix} r_0 & & \cdots & & & r_{n-i+1} \\ & & 0 & w_{s-1} & w_s & \cdots & w_{n-i} \\ & & \iddots & & \vdots & & \vdots \\ & w_{s-1} & & & w_{2s-2} & & w_{n-i+s-2} \\ 0 & & \cdots & 0 & * & \cdots & * \end{pmatrix},$$

where the $*$'s in the last row indicate newly computed entries. This enables one to extend the decomposition to $H_{m-n+i+s,i+s}$ and completes one step of the algorithm.

This step can also be described in terms of polynomial operations. Reducing the right-hand-side of (2.9) to the form given in (2.10) is equivalent to solving

$$\begin{pmatrix} w_{s-1} & & 0 & r_0 \\ \vdots & \ddots & & \vdots \\ w_{2s-1} & \cdots & w_{s-1} & r_s \end{pmatrix} \begin{pmatrix} \hat{v}_0 \\ \vdots \\ \hat{v}_s \\ \hat{u} \end{pmatrix} = 0. \tag{2.11}$$

Written in this form it is easy to see that (2.11) has the polynomial formulation:

$$((w_{s-1} + \ldots + w_{2s-1}z^s)(\hat{v}_0 + \ldots + \hat{v}_s z^s) + \hat{u}R(z)) \pmod{z^{s+1}} = 0$$

or

$$(z^{-s+1}W(z)\hat{V}(z) + \hat{u}R(z)) \pmod{z^{s+1}} = 0. \tag{2.12}$$

From this last, we have that

$$\hat{V} = -(\hat{u}z^{s-1}R(z)W^{-1}(z)) \pmod{z^{s+1}},$$

so we see that the row operations performed to obtain (2.10) are equivalent to the polynomial division of $W(z)$ into $R(z)$. But now we can also express the row operations applied to $L(i+s+1)$ in terms of polynomial multiplication.

If the last row of $L(i+s+1)$ is denoted by $(v^+_{s+i},\ldots,v^+_1,v^+_0)$, then

$$V^+(z) = \hat{u}z^{s+1}Q(z) + \hat{V}(z)V(z).$$

But now, using the correspondence between Hankel systems and the Padé approximant problem, it is clear that the last row of $L(i+s+1)$ is the denominator of the $(m-n+i+s, i+s)$ Padé form for $A(z)/-1$ and the last row of $U(i+s+1)$ is the corresponding residual.

Rissanen's algorithm continues by examining the last row of $U(i+s+1)$ to find its first nonzero entry. Then the process is repeated, but now the last row and the i^{th} row are used to perform the reductions at the next step. (In terms of polynomials, $z^sV(z)$ and $V^+(z)$ are used to continue the process.) Rissanen's algorithm keeps track of the two rows that are required by one step of the algorithm to perform the reductions. It is this book-keeping that enables the algorithm to step over singular principal submatrices of $H_{m,n}$. These singular submatrices correspond to entries in singular blocks in the associated Padé table and the number of rows that are added in one step of the algorithm correspond to the number of such entries that are encountered.

By exploiting the Hankel structure, Rissanen's algorithm reduces the number of operations that need to be performed. It is called a *fast* algorithm as its operation count is $O(n^2)$. There are other fast methods. One of the earliest such methods is the one of Levinson [41] dating back to 1947. This method solved the problem for symmetric Toeplitz systems, though it

was not explicit in its use of matrix methods. Since then, Levinson's method has been extended to the general case and other fast methods have appeared (Trench [57], Zohar [68], [69], Bareiss [4], Jain [34], Delsarte and Genin [20]). All these fast algorithms operate in a similar fashion; they may use different submatrices or phrase their operation in terms of bordering techniques, but they are equivalent to Gaussian elimination without pivoting.

2.3 The Algorithm of Cabay and Choi

We have now seen how the structure of the Hankel matrix can be exploited to reduce the work involved in solving such a system. We emphasized the correspondence with the polynomial representation of the problem. In particular, we saw that in Rissanen's algorithm the row reduction operations performed in one step were equivalent to doing a polynomial division of W into R. This correspondence was used by Cabay and Choi [14] to derive a polynomial algorithm that is equivalent to Rissanen's, but with a better operation count. The improved operation count is achieved by observing that only the first $2s-1$ terms of $W(z)$ and the first s terms of $R(z)$ are needed during any one step; Rissanen's algorithm computes the entire row in the corresponding matrix.

Their development was entirely from a polynomial point of view. The algorithm iteratively computes Padé forms along an arbitrary, but fixed, di-

agonal of the Padé table in an algebraic (i.e., exact arithmetic) setting. To present the algorithm, we introduce some new notation and make a few (non-restrictive) assumptions. The presentation we now give differs considerably from that in [14], but the changes enable us to emphasize those points which are crucial for understanding and presenting the algorithm of the next chapter.

To simplify the notation, we first assume that $m = n - 1$. This is not a real restriction as the following lemma shows.

Lemma 2.8 *If we can obtain an $(n-1, n)$ Padé approximant for an arbitrary power series, then we can obtain an (m, n) Padé approximant for $A(z)$ from an $(n-1, n)$ Padé approximant for a certain power series using only polynomial multiplication and/or the removal of factors of the form z^{α}.*

Proof Suppose, first, that $m < n - 1$. By assumption, we can obtain an $(n-1, n)$ Padé approximant (U, V), for $z^{n-m-1}A(z)$. From the order condition, $z^{n-m-1}AV - U = O(z^{2n})$, it is clear that z^{n-m-1} divides U, so that $U/(z^{n-m-1})$ is a polynomial of degree m. But then,

$$AV - U/z^{n-m-1} = O(z^{2n-(n-m-1)}) = O(z^{m+n+1}),$$

so that $(U/z^{n-m-1}, V)$ is an (m, n) Padé approximant for $A(z)$.

Now suppose that $m > n-1$. We can decompose A as $A = A_1 + z^{m-n+1}A_2$, where $\partial(A_1) = m - n$. By assumption, we can obtain an $(n-1, n)$ Padé

approximant (U, V), for $A_2(z)$. We then have $A_2V - U = O(z^{2n})$, so that

$$\begin{aligned} AV - (A_1V + z^{m-n+1}U) &= A_1V + z^{m-n+1}A_2V - A_1V - z^{m-n+1}U \\ &= z^{m-n+1}(A_2V - U) \\ &= O(z^{2n+m-n+1}) = O(z^{m+n+1}). \end{aligned}$$

Now $\partial(A_1V) = (m-n)+n = m$ and $\partial(z^{m-n+1}U) = (m-n+1)+(n-1) = m$, so that $\partial(A_1V + z^{m-n+1}U) \leq m$ and we see that $(A_1V + z^{m-n+1}U, V)$ is an (m, n) Padé approximant for $A(z)$. ∎

Now we assume that $A(z)$ and $B(z)$ are fixed and we define a sequence of points along the superdiagonal of the Padé table

$$n_0, n_1, n_2, \cdots,$$

by setting $n_0 = 0$ and $n_{i+1} = n_i + s_i$, where the $s_i \geq 1$ are selected so that $S_{n_i} = S_{n_i-1,n_i}$ is nonsingular. (The algorithm of Cabay and Choi actually selects the first such s_i, but we impose no such restriction.)

Because S_{n_i} is nonsingular, both the $(n_i - 2, n_i - 1)$ and $(n_i - 1, n_i)$ Padé forms can be made unique by imposing appropriate conditions. We use the following.

Definition 2.9 *The* **normalized Padé approximant** *of type* (m, n) *for the power series quotient* $A(z)/B(z)$ *is the* (m, n) *Padé form for which* $\| U(z) \| +$

$\| V(z) \| = 1$, *where* $\| \cdot \|$ *is some norm defined for polynomials of arbitrary degree.*

This is not the normalization condition used in [14], but it is the most convenient for our purposes.

We now introduce some notation that allows us to present the algorithm in a concise form. With $n_0, n_1, \ldots$, defined as above, we define $\mathcal{P}^{(i)}$, for $A(z)/B(z)$, to be

$$\mathcal{P}^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and

$$\mathcal{P}^{(i)} = \begin{pmatrix} P^{(i)} & Q^{(i)} \\ U^{(i)} & V^{(i)} \end{pmatrix}, i = 1, 2, \ldots,$$

where $(U^{(i)}, V^{(i)})$ is the $(n_i - 1, n_i)$ normalized Padé approximant and $(P^{(i)}, Q^{(i)})$ is the $(n_i - 2, n_i - 1)$ normalized Padé approximant multiplied by z^2. The residual sequence associated with these Padé forms will be denoted by

$$\left\{\mathcal{R}^{(i)}\right\} = \left\{\begin{pmatrix} R^{(i)} \\ W^{(i)} \end{pmatrix}\right\}, i = 0, 1, \cdots,$$

where

$$\mathcal{P}^{(i)} \begin{pmatrix} B(z) \\ A(z) \end{pmatrix} = z^{2n_i} \begin{pmatrix} R^{(i)} \\ W^{(i)} \end{pmatrix}.$$

If we now let

$$\mathcal{A} = \begin{pmatrix} B(z) \\ A(z) \end{pmatrix}$$

then $\mathcal{P}^{(i)}$ satisfies

$$\mathcal{P}^{(i)}\mathcal{A} = z^{2n_i}\mathcal{R}^{(i)}.$$

We will refer to $\mathcal{P}^{(i)}$ as the n_i^{th} Padé approximant pair for A/B. (We can also define the Padé approximant pair for the case where S_{n_i} is singular; in this case, the Padé approximant pair will be nonunique.) We now develop a method for iteratively producing $\mathcal{P}^{(i)}$.

We can view $W_i(z)/R_i(z)$ as a power series quotient and compute its normalized Padé approximants as well[2]. Let $\hat{S}_{s_i}$ be the Sylvester system associated with the $(s_i - 1, s_i)$ Padé form for W_i/R_i. We define $\hat{\mathcal{P}}^{(i)}$ for W_i/R_i to be

$$\hat{\mathcal{P}}^{(i)} = \begin{pmatrix} 0 & z^2 \\ \hat{U}^{(i)} & \hat{V}^{(i)} \end{pmatrix}, \text{ if } s_i = 1,$$

and

$$\hat{\mathcal{P}}^{(i)} = \begin{pmatrix} \hat{P}^{(i)} & \hat{Q}^{(i)} \\ \hat{U}^{(i)} & \hat{V}^{(i)} \end{pmatrix}, \text{ for } s_i > 1,$$

where $(\hat{U}^{(i)}, \hat{V}^{(i)})$ is a $(s_i - 1, s_i)$ normalized Padé approximant for W_i/R_i and $(\hat{P}^{(i)}, \hat{Q}^{(i)})$ is a $(s_i - 2, s_i - 1)$ normalized Padé approximant for W_i/R_i multiplied by z^2. We will refer to $\hat{\mathcal{P}}^{(i)}$ as the s_i^{th} Padé approximant pair for the residual sequence $\mathcal{R}^{(i)}$.

---

[2]Recall that our selection of s_{i-1} ensures that $r_0^{(i)} \neq 0$.

Lemma 2.10 *If S_{n_i} is nonsingular, then*

$$\bar{\mathcal{P}} = \hat{\mathcal{P}}^{(i)}\mathcal{P}^{(i)}$$

is an n_{i+1}^{th} Padé approximant pair for A/B.

Proof We have that

$$\hat{\mathcal{P}}^{(i)}\begin{pmatrix} R^{(i)} \\ W^{(i)} \end{pmatrix} = z^{2s_i}\begin{pmatrix} \tilde{R} \\ \tilde{W} \end{pmatrix}$$

and

$$\begin{aligned} \bar{\mathcal{P}}\begin{pmatrix} B \\ A \end{pmatrix} &= \hat{\mathcal{P}}^{(i)}\mathcal{P}^{(i)}\begin{pmatrix} B \\ A \end{pmatrix} = \hat{\mathcal{P}}^{(i)}z^{2n_i}\begin{pmatrix} R^{(i)} \\ W^{(i)} \end{pmatrix} \\ &= z^{2n_i}\hat{\mathcal{P}}^{(i)}\begin{pmatrix} R^{(i)} \\ W^{(i)} \end{pmatrix} \\ &= z^{2n_i}z^{2s_i}\begin{pmatrix} \tilde{R} \\ \tilde{W} \end{pmatrix} = z^{2n_{i+1}}\begin{pmatrix} \tilde{R} \\ \tilde{W} \end{pmatrix} \end{aligned} \tag{2.13}$$

and we see that $\bar{\mathcal{P}}$ satisfies the order condition for the $(n_i + s_i - 2, n_i + s_i - 1)$ and $(n_i + s_i - 1, n_i + s_i)$ Padé forms. It is easy to see that the entries of $\bar{\mathcal{P}}$ are polynomials of the appropriate degrees as well. For example, if we let

$$\bar{\mathcal{P}} = \begin{pmatrix} \bar{P} & \bar{Q} \\ \bar{U} & \bar{V} \end{pmatrix},$$

then

$$\bar{Q} = \hat{Q}^{(i)}V^{(i)} + \hat{P}^{(i)}Q^{(i)} \tag{2.14}$$

so that

$$\begin{aligned}\partial(\bar{Q}) &= max\{\partial(\hat{Q}^{(i)}V^{(i)}), \partial(\hat{P}^{(i)}Q^{(i)})\} \\ &\leq max\{((s_i+1)+n_i), (s_i+n_i+1)\} \\ &= s_i+n_i+1.\end{aligned}$$

Now the constant and linear terms of $\bar{Q}$ are zero, so we see that $\bar{Q}$ satisfies the degree conditions for the denominator of a (n_i+s_i-2, n_i+s_i-1) Padé form multiplied by z^2. All that remains to be shown is that $\bar{Q}$ and $\bar{V}$ are nonzero. We show this for $\bar{Q}$; $\bar{V}$ follows by a similar argument. We have that $\bar{Q}$ satisfies (2.14) and $\hat{Q}^{(i)}$ satisfies:

$$\begin{pmatrix} w_0 & & 0 & r_0 & & 0 \\ & \ddots & & & \ddots & \\ \vdots & & & \vdots & & r_0 \\ & & w_0 & & & \vdots \\ \vdots & & \vdots & & & \\ w_{2s_i-2} & & w_{s_i} & r_{2s_i-2} & & r_{s_i} \end{pmatrix} \begin{pmatrix} \hat{q}_2 \\ \vdots \\ \hat{q}_{s_i+1} \\ \hat{p}_2 \\ \vdots \\ \hat{p}_{s_i} \end{pmatrix} = \begin{pmatrix} 0 \\ \\ \vdots \\ 0 \\ \tilde{r}_0 \end{pmatrix}. \qquad (2.15)$$

Suppose now that $\bar{Q} = 0$. By definition of $\hat{P}^{(i)}$ and $\hat{Q}^{(i)}$, we have that $\hat{q}_0 = \hat{q}_1 = \hat{p}_0 = \hat{p}_1 = 0$. Then from (2.14), we have $\hat{q}_2 = \hat{q}_3 = 0$ as $v_0^{(i)} \neq 0$. But then from (2.15) we can conclude that $\hat{p}_2 = 0$ as $r_0^{(i)} \neq 0$. But now we can refer back to (2.14) and conclude that $\hat{q}_4 = 0$. Continuing this argument we see that $\hat{Q}^{(i)} = 0$, which is a contradiction. Therefore, $\bar{Q} \neq 0$. Hence, we have that $(\bar{P}, \bar{Q})$ is a (n_i+s_i-2, n_i+s_i-1) Padé form multiplied by z^2

and $(\bar{U}, \bar{V})$ is a $(n_i + s_i - 1, n_i + s_i)$ Padé form. ∎

This lemma says that by solving the associated residual system for a given Padé approximant, we can obtain a higher order Padé form. However, if the residuals satisfy certain conditions, we can say more than this. The principal result of this section is the following.

Theorem 2.11 *If S_{n_i} is nonsingular then,*

$$\hat{S}_{s_i} \text{ is nonsingular} \Leftrightarrow S_{n_{i+1}} \text{ is nonsingular.}$$

Proof We let $\bar{\mathcal{P}} = \hat{\mathcal{P}}^{(i)}\mathcal{P}^{(i)}$. From Lemma 2.10, we know that $\bar{\mathcal{P}}$ defines a Padé form. If we let $\bar{R}$ and $\bar{W}$ denote the residuals associated with $\bar{\mathcal{P}}$, then from (2.13), we also have

$$z^{2n_{i+1}} \begin{pmatrix} \bar{R} \\ \bar{W} \end{pmatrix} = \bar{\mathcal{P}} \begin{pmatrix} B \\ A \end{pmatrix} = z^{2n_i} \hat{\mathcal{P}}^{(i)} \begin{pmatrix} R^{(i)} \\ W^{(i)} \end{pmatrix} = z^{2n_{i+1}} \begin{pmatrix} \tilde{R} \\ \tilde{W} \end{pmatrix}. \quad (2.16)$$

Suppose first that $\hat{S}_{s_i}$ is nonsingular. Then, from Theorem 2.7, $\hat{v}_0^{(i)} \neq 0$ and $\tilde{r}_0 \neq 0$. But then, $\bar{v}_0 = v_0^{(i)}\hat{v}_0^{(i)} \neq 0$ and, from (2.16), $\bar{r}_0 = \tilde{r}_0 \neq 0$, so that by Theorem 2.7 $S_{n_i+s_i}$ is nonsingular.

Now suppose that $S_{n_{i+1}}$ is nonsingular. But then, by Theorem 2.7, $\bar{v}_0 \neq 0$ and $\bar{r}_0 \neq 0$, so that $\hat{v}_0^{(i)} \neq 0$ and $\tilde{r}_0 \neq 0$. But, again applying Theorem 2.7, we have that $\hat{S}_{s_i}$ is nonsingular. ∎

Now Lemma 2.10 and Theorem 2.11 allow us to generate $\mathcal{P}^{(i)}$ iteratively as the following Corollary indicates.

Corollary 2.12 *If S_{n_i} is nonsingular, then, with α and β appropriately chosen nonzero scalar values,*

$$\mathcal{P}^{(i+1)} = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix} \hat{\mathcal{P}}^{(i)} \mathcal{P}^{(i)},$$

gives us the $(n_i + s_i - 1, n_i + s_i)$ normalized Padé approximant for A/B and the $(n_i + s_i - 2, n_i + s_i - 1)$ normalized Padé approximant for A/B multiplied by z^2.

Proof This is immediate from Lemma 2.10 and Theorem 2.11. ∎

These results define a three-term recurrence relation among the entries along the superdiagonal of the Padé table. They are actually valid for any (m, n) and so define a three-term recurrence relation among the entries along any off-diagonal of the Padé table.

As remarked earlier, the fast algorithm of Cabay and Choi selects s_i so that it is the first stepsize s_i for which $\hat{S}_{s_i}$ is nonsingular. In this case, the

residual system to be solved has the form:

$$\begin{pmatrix} 0 & 0 & r_0 & 0 \\ \vdots & \vdots & & \ddots \\ w_{s_i-1} & & \vdots & r_0 \\ & \ddots\ 0 & & \\ w_{2s_i-2} & w_{s_i-1} & r_{2s_i-1} & r_{s_i} \end{pmatrix} \begin{pmatrix} \hat{v}_1 \\ \vdots \\ \hat{v}_{s_i} \\ \hat{u}_0 \\ \vdots \\ \hat{u}_{s_i-1} \end{pmatrix} = -\hat{v}_0 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ w_{s_i-1} \\ \vdots \\ w_{2s_i-1} \end{pmatrix}. \tag{2.17}$$

But, easily, $\hat{u}_0 = \cdots = \hat{u}_{s_i-2} = 0$, so (2.17) reduces to (2.11), which means that it can be solved by a polynomial division. Also, it is easy to show that $(\hat{P}^{(i)}, \hat{Q}^{(i)}) = (0, z^{s+1})$. Lemma 2.10 is then applied to generate $\mathcal{P}^{(i+1)}$ and Theorem 2.11 allows the algorithm to continue iteratively down the superdiagonal of the Padé table.

Other relationships exist between entries in the Padé table and they yield other algorithms (Brent et al [6], Claessens [15], Claessens and Wuytack [16], Bultheel [9]). Some are diagonal algorithms like that of Cabay and Choi, other are recursive like that of Brent et al, and still others operate along the anti-diagonals of the Padé table (or other patterns in the table) to compute a given Padé approximant. By selecting a larger stepsize and exploiting fast polynomial multiplication, the algorithm of Cabay and Choi can be modified to yield an iterative recursive "superfast" algorithm that is $O(n \log^2 n)$. In this form, the algorithm of Brent et al [6] is equivalent to that of Cabay and

Choi, though their algorithm starts from observing the relationship between Padé approximants and the extended fast Euclidean algorithm. With this formulation, unfortunately, the algorithm that results is not iterative, so the order of the desired Padé approximant has to be known *a priori*. Some of the other $O(n \log^2 n)$ superfast algorithms are Bitmead and Anderson [5], Kumar [38], de Hoog [19], Ammar and Gragg [1]. All these algorithms are illustrative of the approaches that are based on the Padé table as opposed to the Hankel (or Toeplitz) formulation of the problem.

The important observation to be made about the above is that Theorem 2.11 is valid for any s_i for which $\hat{S}_{s_i}$ is nonsingular, not just the first such (as used by Cabay and Choi). It is this fact that forms the basis for the algorithm described in the next chapter.

Chapter 3

Numerical Padé Approximant Algorithm

In this chapter, we present an algorithm that numerically computes a Padé approximant in a stable fashion. It does so by computing a sequence of approximants of increasing order. These approximants correspond to well-conditioned linear systems and are selected by the application of a simple condition. In this fashion, the errors introduced by the algorithm through the rounding process are guaranteed to be small. In Chapters 4 and 5, we provide the analytic justification for this assertion and in Chapter 6 we provide experimental results showing that the condition can be satisfied in practice. We start this chapter by briefly examining the stability properties of the algorithms presented in the previous chapter and then give a short

discussion about stability.

3.1 Numerical Considerations

All the algorithms presented in the previous chapter were algebraic; that is, they all assumed exact arithmetic. In practice, this is seldom possible, though there are implementations of various algorithms using symbolic systems (see, for example, Geddes [24], Kossowski [37]). As mentioned in the discussion of Rissanen's algorithm, some algorithms fail if singular submatrices are encountered (see Claessen [15]). From the Padé approximation point of view these algorithms are failing because singular blocks have been encountered in the Padé table. Rissanen's and Choi's algorithm handle the singular cases by recognizing them and then using special rules (i.e., by taking larger steps).[1] But problems arise if the near-singular case is encountered. For example, in Rissanen's algorithm we could reach a stage where w_0 is nonzero, but very small relative to r_0 and $w_1, w_2, \cdots$. Then a step of size one is taken by the algorithm, but considered in the context of Gaussian elimination, this is equivalent to performing a column reduction with an extremely small pivot; this is known to be an unstable operation and poor solutions can result. Other algorithms fail for the same sort of reasons. (See, for example, Bultheel and Wuytack [10], Bunch [11], Luk and Qiao [42].)

[1] Claessens and Wuytack [16], and Bultheel [9] give other such methods.

Bunch [11] examines a number of methods for Toeplitz systems. If the matrix is positive-definite symmetric, the methods perform well numerically, but in the general case they can perform poorly. (Similar observations are made in Bultheel and Wuytack [10].) The observation to be made is that the positive-definite symmetric matrices are a class that can be solved stably using Gaussian elimination without pivoting. Bunch suggests that in order to make these algorithms stable some pivoting must be introduced, but then this would result in the loss of the regular structure of the matrices that enables the algorithms to perform in a fast manner.

Sweet [55] takes a different approach using orthogonal transformations to produce a fast QR factorization of a Toeplitz matrix. Again, the regular structure of the Toeplitz matrix is exploited, but even this more stable technique has its problems. Luk and Qiao [42] give an example of a matrix for which Sweet's factorization produces poor results.

In the last section of this chapter, we describe an algorithm that uses Gaussian elimination with limited pivoting. It introduces just enough pivoting to remain stable. If the Padé approximant problem is such that all principal submatrices of the associated Hankel matrix are well-conditioned, then the algorithm reduces to the algorithm of Cabay and Choi, but if not, then it increases its complexity/operation count just enough to deal with the unstable subproblems as they are encountered.

We previously saw that if $(r_0 v_0)$ was nonzero, then $S_{m,n}$ was nonsingular

(Theorem 2.7). We will see, that with the imposition of certain normalization conditions, the size of $|r_0 v_0|$ also determines the condition number of $S_{m,n}$. By ensuring that $|r_0 v_0|$ is sufficiently large, then only those Padé approximants corresponding to well-conditioned $S_{m,n}$ are computed. In practice, this means that unstable blocks in the Padé table are detected and the stepsize is chosen so that the algorithm steps across the unstable blocks. Before presenting the algorithm, we first give a short discussion about stability in general and introduce some new definitions.

3.2 Stability

In this section we review some of the concepts regarding problem conditioning and algorithmic stability. Given the natural correspondence between the Padé approximant problem and matrix analysis, much of the discussion focuses on matrices. We assume in what follows that we have a matrix norm that is consistent (viz, $\| AB \| \leq \| A \| \| B \|$).

Definition 3.1 *If a square matrix M is nonsingular, then the* **condition number**, $\kappa(M)$, *of M is given by*

$$\kappa(M) = \| M \| \| M^{-1} \|$$

If M is singular, then $\kappa(M) = \infty$.

The condition number of a matrix provides a measure of how sensitive the problem $Mx = b$ is to perturbations in M and b. For example (see Golub and van Loan, [27, pages 81-82]),

Theorem 3.2 *If $Mx = b$ and $(M + \delta M)(x + \delta x) = (b + \delta b)$, where M is nonsingular and if*

$$\frac{\| \delta M \|}{\| M \|} < \frac{1}{\kappa(M)},$$

then $(M + \delta M)$ is nonsingular and

$$\frac{\| \delta x \|}{\| x \|} \leq \frac{\kappa(M)}{1 - \kappa(M)\frac{\| \delta M \|}{\| M \|}} \left\{ \frac{\| \delta M \|}{\| M \|} + \frac{\| \delta b \|}{\| b \|} \right\}.$$

In practice, if $\kappa(M)$ is small, we say that M is well-conditioned. If this is the case, then we know that if we obtain a solution x_c to $Mx = b$ where $\| b - Mx_c \|$ is small, then $\| x_c - x \|$ is small as well. If $\kappa(M)$ is large, then the size of $\| b - Mx_c \|$ tells us nothing about the size of $\| x_c - x \|$. In this case, we say that the problem is ill-conditioned. For ill-conditioned problems, solutions can be extremely sensitive to perturbations introduced during the formulation of the problem.

Once an algorithm is given for solving a numerical problem, the question arises as to how accurately it computes its solution. In other words, how much effect do the errors introduced by the algorithm have on the computed solution. Ideally, we would like them to have minimal effect, but, from our comments about problem conditioning, it is clear that the effect on the

computed solution will also depend upon the condition of the underlying problem. To separate the effects of the algorithm from the effects of the problem's condition, we have the following common definition of algorithmic stability (Bunch [12], Wilkinson [61]):

Definition 3.3 *An algorithm for solving linear equations is* **stable** *for a class of matrices $\mathcal{M}$ if for each M in $\mathcal{M}$ and for each b the computed solution x_c to $Mx = b$ satisfies $\hat{M}x_c = \hat{b}$, where $\hat{M}$ is close[2] to M and $\hat{b}$ is close to b.*

The extension to problems outside matrix analysis is straightforward. In the case of the Padé approximation problem we have:

Definition 3.4 *An algorithm for solving the (m,n) Padé approximant problem is stable if for each pair of power series A and B, the algorithm computes U and V that are the (m,n) Padé approximant for $\hat{A}$ and $\hat{B}$ where $\hat{A}$ is close to A and $\hat{B}$ is close to B, modulo z^{m+n+1}.*

Bunch also introduces the concept of strong stability in [12]:

Definition 3.5 *An algorithm for solving linear equations is* **strongly stable** *for a class of matrices $\mathcal{M}$ if for each M in $\mathcal{M}$ and for each b the computed solution x_c to $Mx = b$ satisifies $\hat{M}x_c = \hat{b}$, where $\hat{M}$ is close to M and $\hat{b}$ is close to b and, moreover, $\hat{M}$ is in the class $\mathcal{M}$.*

[2] An exact meaning for *close* is not given in this definition; what is acceptable depends upon the precise problem that is being solved and the accuracy that is desired.

This is a stronger condition since strong stability implies stability, but not conversely. It may be of interest in those cases where the matrix problem has a physical interpretation. For example (Bunch [12]), a physical problem modelled by partial differential equations may result in a mathematical formulation that requires the solution of a positive-definite symmetric system. If we know that the computed solution is the exact solution to a nearby positive-definite symmetric system, then we may be able to conclude that the computed solution corresponds to some nearby physical system.

If we restate Definition 3.4 in terms of the matrix notation, we require that the computed U and V satisfy

$$\begin{pmatrix} \hat{a}_0 & & 0 & \hat{b}_0 & & 0 \\ & \ddots & & & \ddots & \\ \vdots & & \hat{a}_0 & \vdots & & \hat{b}_0 \\ \vdots & & \vdots & \vdots & & \vdots \\ & & \vdots & & & \vdots \\ \hat{a}_{m+n} & \cdots & \hat{a}_m & \hat{b}_{m+n} & \cdots & \hat{b}_n \end{pmatrix} \begin{pmatrix} v_0 \\ \vdots \\ v_n \\ u_0 \\ \vdots \\ u_m \end{pmatrix} = 0, \qquad (3.1)$$

where the matrix in (3.1) is close to the $\bar{S}_{m,n}$ of (2.2). Since the above matrix is a Sylvester matrix, we see that if the algorithm is stable as defined by Definition 3.4, then the algorithm is also strongly stable over the class of Sylvester matrices. Bunch suggests in [12, page 64] that showing this for the class of Toeplitz matrices would be a "formidable task indeed".

But suppose now that $B(z) \equiv -1$, so we have

$$AV - U = \delta W + O(z^{m+n+1}).$$

Since $v_0 \neq 0$, we can formally produce the power series

$$\delta A = -V^{-1}\delta W.$$

Then

$$(A + \delta A)V - U = O(z^{m+n+1}),$$

so we see that (U, V) solves the perturbed problem $(A + \delta A, -1)$. Unfortunately, even if δW is small, $\delta A = -V^{-1}\delta W$ need not be small. But we do observe that if $|v_0| > |v_i|$, $1 \leq i \leq n$, then δA is small if δW is. This suggests that the size of $|v_0|$ may be a factor in assessing how good the computed Padé approximant is. We will see in later chapters that this is the case.

Now users often really want to obtain solutions that are close to the true solution; they may not need (or want) to know that their solution is the exact solution of a nearby problem. As we have seen, the final solution is dependent not only on the algorithm, but also upon the condition of the problem being solved. If the problem is poorly conditioned, then we cannot expect the computed solution to be good because just formulating the problem numerically will introduce perturbations. But if the problem is well-conditioned, then we would like an accurate solution. To describe this, a weaker concept of stability was also introduced by Bunch in [12].

Definition 3.6 *An algorithm for solving linear equations is* **weakly stable** *for a class of matrices $\mathcal{M}$ if for each well-conditioned M in $\mathcal{M}$ and for each b, the computed solution x_c to $Mx = b$ is such that $\| x - x_c \| / \| x \|$ is small.*

In Cybenko [18], a controversy over the stability of the Levinson-Durbin algorithm for the solution of Toeplitz systems was addressed. Some researchers were claiming stability for the algorithm, while others were claiming instability. In the end analysis, the claims of instability were unsubstantiated due to the fact that the problems that purportedly demonstrated the instability of the algorithm were themselves ill-conditioned. Cybenko then showed for the class of positive-definite symmetric Toeplitz matrices that if the matrix was well-conditioned, then the Levinson-Durbin algorithm produced a solution with a small residual; in other words, he had shown the weak stability of the algorithm for this class of Toeplitz matrices, though he did not actually use this terminology[3]. For these well-conditioned matrices, he obtained analytic results that were comparable to the results obtained from an analysis of the Cholesky algorithm.

The purpose for which Bunch [11], [12] introduced the various concepts of stability (presented above), was to address the confusion about the stability of Toeplitz solvers. What he showed was that the algorithms he considered required that various submatrices be well-conditioned. A submatrix can

[3]Bunch [12] first observed that Cybenko's stability result was actually a proof of weak stability.

be ill-conditioned even though the matrix itself is well-conditioned. The well-conditioning of principal submatrices, however, can be guaranteed if the matrix is positive-definite symmetric. Thus, he concluded, the algorithms were weakly stable for the class of positive-definite symmetric matrices, but were not weakly stable for more general classes. We will show that our algorithm has no such restrictions. We will show, in the course of the next two chapters, that our algorithm is weakly stable for the general class of problems.

3.3 The Algorithm

In the previous section, the definitions and results were given in terms of finding the $(n-1, n)$ Padé approximant for the general $A(z)/B(z)$. For the remainder of this chapter and the chapters that follow, we restrict the power series to the case $B(z) \equiv -1$.[4]

The other assumption that we make with respect to the power series $A(z)$ is that the norm of $A(z) \pmod{z^{2N}}$ is 1. In practice, we only need the first $2N$ terms of $A(z)$ to calculate the $(N-1, N)$ Padé approximant. This normalization simplifies the analysis that is presented in the later chapters. In practice, the implementation of the algorithm handles this detail for the

[4]This is no restriction in a theoretical sense, as the general problem can be converted to this form by multiplying both power series by $-B^{-1}(z)$; that this inverse exists follows from the fact that $B(0) \neq 0$.

user, but it has been omitted from the presentation of the algorithm in order to make the presentation clearer.

With these assumptions, we are now ready to state the algorithm. One thing to keep in mind during the presentation of the algorithm and its subsequent analysis is the definition of $\hat{P}^{(k)}$ and $\hat{Q}^{(k)}$ in $\hat{\mathcal{P}}^{(k)}$. The constant and linear terms in these polynomials are always zero as they are the $(n-2, n-1)$ normalized Padé approximant multiplied by z^2. This enables us to present the algorithm in a much more compact fashion.

Algorithm: NPADE

Input: $A(z)$, N, ε, where

N is a positive integer, $N \geq 1$,
$A(z)$ is a real power series (only $A \pmod{z^{2N}}$ is needed),
and ε is a real value, $0 \leq \varepsilon < 1$.

Output: The $(n_k - 1, n_k)$ normalized Padé approximant for $A(z)$, where $n_k \leq N$ is the largest n_k for which the normalized Padé approximant can be computed stably. If $n_k < N$, then an $(N-1, N)$ normalized Padé approximant is also output.

Program:

/* Initialization */

$i \leftarrow 0$;

$n_i \leftarrow 0$;

$s_i \leftarrow 1$;

$\mathcal{P}^{(i)} \leftarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$;

/* Compute the Padé Fraction */

While $(n_i + s_i \leq N)$ do

Compute $\mathcal{R}^{(i)}$ such that $(\mathcal{P}^{(i)}\mathcal{A}) mod(z^{2n_i+2s_i}) = z^{2n_i}\mathcal{R}^{(i)}$;

Assemble the $2s_i \times 2s_i$ Sylvester matrix $\hat{S}_i$;

Solve the Sylvester system to obtain $\hat{\mathcal{P}}^{(i)}$;

$\mathcal{P}^{(i+1)} = \hat{\mathcal{P}}^{(i)}\mathcal{P}^{(i)}$;

Rescale $\mathcal{P}^{(i+1)}$ and $\hat{\mathcal{P}}^{(i)}$ so that $\| P^{(i+1)} \| + \| Q^{(i+1)} \| = 1$ and $\| U^{(i+1)} \| + \| V^{(i+1)} \| = 1$;

Compute $r_0^{(i+1)}$ such that $((\mathcal{P}^{(i+1)}\mathcal{A}) mod(z^{2n_i+2s_i+1}))_1 = z^{2n_i+2s_i} r_0^{(i+1)}$;

if $|r_0^{(i+1)} v_0^{(i+1)}| > \varepsilon$ then

$n_{i+1} \leftarrow n_i + s_i$;

$i \leftarrow i + 1$;

$s_i \leftarrow 1$;

else

$s_i \leftarrow s_i + 1$;

End while;

Output $(U^{(i)}, V^{(i)})$;

/* Check the Order of $(U^{(i)}, V^{(i)})$ */

If $(n_i < N)$ then

/* Output an $(N-1, N)$ Padé approximant */

$n_{i+1} \leftarrow N$;

$i \leftarrow i + 1$;

Output $(U^{(i)}, V^{(i)})$;

This completes the statement of the algorithm. The algorithm requires the solution of a Sylvester system. This is done using Gaussian elimination. If the Sylvester system is singular, the solver is assumed to generate some solution for which $\hat{V}^{(i)}$ and $\hat{Q}^{(i)}$ are nonzero. In this case, $|r_0^{(i+1)} v_0^{(i+1)}|$ will be zero and the solution will be rejected. Also, there may be a better way to obtain s_i. As the algorithm is stated, increasing values of s_i are successively tried until one is found that satisfies the acceptance criteria. There may be a way to select s_i without all these computations (perhaps directly from $R^{(i)}$ and $W^{(i)}$), but none has been found yet. However, the point must be made that if the unstable blocks are small, then the current method has no great effect on the cost.

If the While loop is exited with $n_i = N$, then the normalized Padé approximant will have a small residual and be close to the true normalized Padé approximant. If the While loop is exited with $n_i < N$, then a $(N-1, N)$ normalized Padé approximant is computed outside the loop anyway. If the final system that is solved is nonsingular, then the computed normalized Padé approximant will have a small residual, but we make no claims that it will be close to the true normalized Padé approximant. If the last system that is solved is singular, then no claims are made about the computed normalized Padé approximant or its residual.

By assuming exact arithmetic and using $\epsilon = 0$, the algorithm is equivalent to the Cabay and Choi algorithm presented in Section 2.3. The next two

chapters are devoted to analysing the situation where the arithmetic is not exact.

Chapter 4

Hankel Matrices and Their Inverses

Given exact Padé approximants, there exist various formulae that express the inverse of the associated Hankel and Toeplitz systems in terms of these approximants (see, for example, Gohberg and Semencul [26] and Labahn et al [39]). In this chapter, we derive one such formula for the situation where computed, rather than exact, approximants are used. What we actually obtain is an expression that is almost the inverse of the associated Hankel system. From this expression, we can obtain bounds on the condition number of the system. These bounds depend upon how closely the computed approximants satisfy the order condition defining the Padé approximants. If the error is small, then the most important factor affecting the condition number of the

Hankel system is $1/|r_0 v_0|$.

4.1 Inverses

We momentarily drop one of the assumptions made prior to the presentation of the algorithm. In particular, we now assume that we have obtained the more general (m,n) Padé form (U,V) and the $(m-1,n-1)$ Padé form (P,Q) to the power series $A(z)$.[1] These are computed approximants and so we have

$$AQ - P = \delta R + z^{m+n-1} R \tag{4.1}$$

and

$$AV - U = \delta W + z^{m+n+1} W, \tag{4.2}$$

where $v_0 \neq 0$ and $r_0 \neq 0$. δR and δW are the errors in the order condition. Multiplying (4.2) by Q and (4.1) by V and then subtracting (4.2) from (4.1) gives us

$$QU - VP = V\delta R - Q\delta W + z^{m+n-1}(VR - z^2 QW). \tag{4.3}$$

Now $\partial(QU - VP) \leq m+n-1$, so all higher terms on the right-hand-side of (4.3) must cancel. What we have is

$$QU - VP = \vartheta(z) + v_0 r_0 z^{m+n-1}, \tag{4.4}$$

[1] P and Q are not multiplied by z^2 as in the definition of $\hat{\mathcal{P}}^{(k)}$.

where $\vartheta(z) = (V\delta R - Q\delta W) mod(z^{m+n})$. Using (4.4), it is easy to see that

$$\begin{pmatrix} u_m & \cdots & u_{m-n+1} \\ & \ddots & \vdots \\ 0 & & u_m \end{pmatrix} \begin{pmatrix} q_{n-1} & \cdots & q_0 \\ & \ddots & \vdots \\ 0 & & q_{n-1} \end{pmatrix}$$

$$- \begin{pmatrix} p_{m-1} & \cdots & p_{m-n} \\ & \ddots & \vdots \\ 0 & & p_{m-1} \end{pmatrix} \begin{pmatrix} v_n & \cdots & v_1 \\ & \ddots & \vdots \\ 0 & & v_n \end{pmatrix}$$

$$= r_0 v_0 I + \begin{pmatrix} \vartheta_{m+n-1} & \cdots & \vartheta_m \\ & \ddots & \vdots \\ 0 & & \vartheta_{m+n-1} \end{pmatrix}$$

$$= r_0 v_0 I + [\vartheta_{m+n-1+i-j}]_{i,j=1,\cdots,n}. \qquad (4.5)$$

Also, it is easy to show that

$$\begin{pmatrix} 0 & & v_n \\ & \iddots & \vdots \\ v_n & \cdots & v_1 \end{pmatrix} \begin{pmatrix} q_{n-1} & \cdots & q_0 \\ & \ddots & \vdots \\ 0 & & q_{n-1} \end{pmatrix} - \begin{pmatrix} 0 & & q_{n-1} \\ & \iddots & \vdots \\ q_{n-1} & \cdots & q_0 \end{pmatrix} \begin{pmatrix} v_n & \cdots & v_1 \\ & \ddots & \vdots \\ 0 & & v_n \end{pmatrix}$$

$$= 0. \qquad (4.6)$$

With these observations we then obtain the following lemma.

Lemma 4.1

$$H_{m,n}\begin{pmatrix} v_{n-1} & \cdots & v_0 \\ \vdots & \iddots & \\ v_0 & & 0 \end{pmatrix}$$

$$= \begin{pmatrix} u_m & \cdots & u_{m-n+1} \\ & \ddots & \vdots \\ 0 & & u_m \end{pmatrix} - H_{m-n,n}\begin{pmatrix} 0 & & v_n \\ & \iddots & \vdots \\ v_n & \cdots & v_1 \end{pmatrix}$$

$$+ [\delta w_{m+i-j}]_{i,j=1,\cdots,n} \qquad (4.7)$$

where

$$H_{m,n} = \begin{pmatrix} a_{m-n+1} & \cdots & a_m \\ \vdots & & \\ a_m & \cdots & a_{m+n-1} \end{pmatrix}.$$

Proof By equating the coefficients of z^{m+i-j} in equation (4.2), we obtain

$$(a_{m-n+i-j}v_n + \cdots + a_{m-n+i-1}v_{n+1-j}) + (a_{m-n+i}v_{n-j} + \cdots + a_{m+i-j}v_0)$$

$$- u_{m+i-j}$$

$$= \delta w_{m+i-j}.$$

Using the fact that $v_l = 0$ if $l < 0$ or $l > n$, this can be written as:

$$(a_{m-2n+i}v_{2n-j} + \cdots + a_{m-n+i-1}v_{n+1-j}) + (a_{m-n+i}v_{n-j} + \cdots + a_{m-1+i}v_{0-j+1})$$

$$- u_{m+i-j}$$

$$= \delta w_{m+i-j}$$

or, equivalently,

$$a_{m-n+i}v_{n-j} + \cdots + a_{m-1+i}v_{0-j+1} \tag{4.8}$$
$$= u_{m+i-j} - (a_{m-2n+i}v_{2n-j} + \cdots + a_{m-n+i-1}v_{n+1-j}) + \delta w_{m+i-j}.$$

But (4.8) is the equation obtained by equating the (i,j)-th entries in (4.7). Since $1 \le i,j \le n$, (4.7) holds and the result is proved. ∎

By repeating the above argument for the coefficients of $z^{m+i-j-1}$, $1 \le i,j \le n$, in (4.1), we also obtain:

Lemma 4.2

$$H_{m,n}\begin{pmatrix} q_{n-2} & \cdots & q_0 & 0 \\ \vdots & \iddots & \iddots & \vdots \\ q_0 & \iddots & & \vdots \\ 0 & \cdots & \cdots & 0 \end{pmatrix}$$
$$= \begin{pmatrix} p_{m-1} & \cdots & p_{m-n} \\ & \ddots & \vdots \\ 0 & & p_{m-1} \end{pmatrix} - H_{m-n,n}\begin{pmatrix} 0 & & q_{n-1} \\ & \iddots & \vdots \\ q_{n-1} & \cdots & q_0 \end{pmatrix}$$
$$+ [\delta r_{m+i-j-1}]_{i,j=1,\cdots,n}. \tag{4.9}$$

These two lemmas enable us to prove the following theorem which says that if the errors in the order conditions are small, then we can use $Q(z)$ and $V(z)$

to construct an approximate inverse to $H_{m,n}$.[2]

Theorem 4.3

$$H_{m,n}\left\{\begin{pmatrix} v_{n-1} & \cdots & v_0 \\ \vdots & \iddots & \\ v_0 & & 0 \end{pmatrix}\begin{pmatrix} q_{n-1} & \cdots & q_0 \\ & \ddots & \vdots \\ 0 & & q_{n-1} \end{pmatrix}\right.$$
$$\left.-\begin{pmatrix} q_{n-2} & \cdots & q_0 & 0 \\ \vdots & \iddots & \iddots & \vdots \\ q_0 & \iddots & & \vdots \\ 0 & \cdots & \cdots & 0 \end{pmatrix}\begin{pmatrix} \ddots & & \\ v_n & \cdots & v_1 \\ & \ddots & \vdots \\ 0 & & v_n \end{pmatrix}\right\}$$

$$= r_0 v_0 I + [\vartheta_{m+n-1+i-j}] \tag{4.10}$$
$$+ [\delta w_{m+i-j}]\begin{pmatrix} q_{n-1} & \cdots & q_0 \\ & \ddots & \vdots \\ 0 & & q_{n-1} \end{pmatrix} - [\delta r_{m+i-j-1}]\begin{pmatrix} v_n & \cdots & v_1 \\ & \ddots & \vdots \\ 0 & & v_n \end{pmatrix}.$$

Proof Using (4.7), (4.9), and (4.6),

$$\ddot{H}_{m,n}\left\{\begin{pmatrix} v_{n-1} & \cdots & v_0 \\ \vdots & \iddots & \\ v_0 & & 0 \end{pmatrix}\begin{pmatrix} q_{n-1} & \cdots & q_0 \\ & \ddots & \vdots \\ 0 & & q_{n-1} \end{pmatrix}\right.$$

[2] We use $[a_{f(i,j)}]$ to denote $[a_{f(i,j)}]_{i,j=1,\ldots,n}$, where $a_{f(i,j)} = 0$ if $f(i,j) < 0$

$$
- \left(\begin{pmatrix} q_{n-2} & \cdots & q_0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ q_0 & \ddots & & \vdots \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \begin{pmatrix} v_n & \cdots & v_1 \\ & \ddots & \vdots \\ 0 & & v_n \end{pmatrix} \right\}
$$

$$
= \left\{ \left(\begin{pmatrix} u_m & \cdots & u_{m-n+1} \\ & \ddots & \vdots \\ 0 & & u_m \end{pmatrix} - H_{m-n,n} \begin{pmatrix} 0 & & v_n \\ & \ddots & \vdots \\ v_n & \cdots & v_1 \end{pmatrix} + [\delta w_{m+i-j}] \right\}
$$

$$
\cdot \begin{pmatrix} q_{n-1} & \cdots & q_0 \\ & \ddots & \vdots \\ 0 & & q_{n-1} \end{pmatrix}
$$

$$
- \left\{ \left(\begin{pmatrix} p_{m-1} & \cdots & p_{m-n} \\ & \ddots & \vdots \\ 0 & & p_{m-1} \end{pmatrix} - H_{m-n,n} \begin{pmatrix} 0 & & q_{n-1} \\ & \ddots & \vdots \\ q_{n-1} & \cdots & q_0 \end{pmatrix} + [\delta r_{m+i-j-1}] \right\}
$$

$$
\cdot \begin{pmatrix} v_n & \cdots & v_1 \\ & \ddots & \vdots \\ 0 & & v_n \end{pmatrix}
$$

$$
= \begin{pmatrix} u_m & \cdots & u_{m-n+1} \\ & \ddots & \vdots \\ 0 & & u_m \end{pmatrix} \begin{pmatrix} q_{n-1} & \cdots & q_0 \\ & \ddots & \vdots \\ 0 & & q_{n-1} \end{pmatrix}
$$

$$
- \begin{pmatrix} p_{m-1} & \cdots & p_{m-n} \\ & \ddots & \vdots \\ 0 & & p_{m-1} \end{pmatrix} \begin{pmatrix} v_n & \cdots & v_1 \\ & \ddots & \vdots \\ 0 & & v_n \end{pmatrix}
$$

$$
+ [\delta w_{m+i-j}] \begin{pmatrix} q_{n-1} & \cdots & q_0 \\ & \ddots & \vdots \\ 0 & & q_{n-1} \end{pmatrix} - [\delta r_{m+i-j-1}] \begin{pmatrix} v_n & \cdots & v_1 \\ & \ddots & \vdots \\ 0 & & v_n \end{pmatrix}.
$$

But using (4.5), this last expression is equal to

$$r_0 v_0 I + [\vartheta_{m+n-1+i-j}] + [\delta w_{m+i-j}][q_{n-1+i-j}] - [\delta r_{m+i-j-1}][v_{n+i-j}],$$

which is what we want. ∎

By letting

$$
\tilde{H}_{m,n} = \begin{pmatrix} v_{n-1} & \cdots & v_0 \\ \vdots & \iddots & \\ v_0 & & 0 \end{pmatrix} \begin{pmatrix} q_{n-1} & \cdots & q_0 \\ & \ddots & \vdots \\ 0 & & q_{n-1} \end{pmatrix}
$$

$$
- \begin{pmatrix} q_{n-2} & \cdots & q_0 & 0 \\ \vdots & \iddots & \iddots & \vdots \\ q_0 & \iddots & & \vdots \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \begin{pmatrix} v_n & \cdots & v_1 \\ & \ddots & \vdots \\ 0 & & v_n \end{pmatrix}, \qquad (4.11)
$$

and

$$E = [\vartheta_{m+n-1+i-j}] + [\delta w_{m+i-j}][q_{n-1+i-j}] - [\delta r_{m+i-j-1}][v_{n+i-j}], \qquad (4.12)$$

we can restate the above theorem as

$$H_{m,n}(\frac{1}{r_0 v_0}\tilde{H}_{m,n}) = I + \frac{1}{r_0 v_0}.E \tag{4.13}$$

Intuitively, if $\| \frac{1}{r_0 v_0} E \|$ is small, then $\frac{1}{r_0 v_0}\tilde{H}_{m,n}$ is close to the inverse of $H_{m,n}$ in a certain sense. We make this idea more precise now.

Lemma 4.4 *If* $\| \frac{1}{r_0 v_0} E \| < 1$, *then*

$$H_{m,n}^{-1} = (\frac{1}{r_0 v_0}\tilde{H}_{m,n})(I + \frac{1}{r_0 v_0}E)^{-1} \tag{4.14}$$

and

$$\| (I + \tfrac{1}{r_0 v_0}E)^{-1} \| \leq \frac{1}{1 - \| \frac{1}{r_0 v_0} E \|}. \tag{4.15}$$

Proof Since $\| \frac{1}{r_0 v_0} E \| < 1$, then $I + \frac{1}{r_0 v_0}E$ is invertible (Stewart [54, page 187]) and (4.14) follows from (4.13). A proof of (4.15) can be found in Stewart [54, page 187]. ∎

Thus we see that if $\| \frac{1}{r_0 v_0} E \|$ is small, then $\frac{1}{r_0 v_0}\tilde{H}_{m,n}$ is relatively close to $H_{m,n}^{-1}$. In the absence of errors in the Padé approximants, we have the following.

Corollary 4.5

$$H_{m,n}(\frac{1}{r_0 v_0}\tilde{H}_{m,n}) = I. \tag{4.16}$$

This result first appeared in Labahn et al [39]. Similar formulae appear in Iohvidov [32].

This now gives us an expression for the condition number of the Hankel system. In the next chapter, we look at how errors are introduced in δR and δW and how they grow. We will see that $1/|r_0 v_0|$ is important in describing this error in the order condition. At the end of the chapter, we combine the new results with the results of this chapter to produce expressions for and bounds on the actual errors in the computed normalized Padé approximants.

Chapter 5

Analysis of the Algorithm

The previous chapter showed us that $|r_0 v_0|$ provides information about the nature of the condition number of the Hankel system associated with the Padé approximant problem. How accurate this information is depends upon the sizes of δR and δW in (4.1) and (4.2). Ultimately, though, we are interested in how close the computed normalized Padé approximant (U, V) is to the true normalized Padé approximant. We shall denote the error in this computed Padé approximant by $(\delta U, \delta V)$. In a like fashion, we define $(\delta P, \delta Q)$ to be the error in the computed (P, Q). Now, from (1.10) and (4.2), we see that

$$\begin{pmatrix} a_{m-n+1} & \cdots & a_m \\ \vdots & & \vdots \\ a_m & \cdots & a_{m+n-1} \end{pmatrix} \begin{pmatrix} v_n \\ \vdots \\ v_1 \end{pmatrix} = -v_0 \begin{pmatrix} a_{m+1} \\ \vdots \\ a_{m+n} \end{pmatrix} + \begin{pmatrix} \delta w_{m+1} \\ \vdots \\ \delta w_{m+n} \end{pmatrix},$$

and from this we can obtain an expression for δV in terms of δW. Then (1.9), which defines $U(z)$ in terms of $V(z)$, can be used to obtain δU. Similarly, we can obtain expressions for δP and δQ in terms of δR. Thus, we can express the error in the computed solutions in terms of the errors in the order conditions for the normalized Padé approximants.

This chapter presents an analysis that first gives expressions and bounds for δR and δW. The majority of this chapter is devoted to this analysis. A forward analysis type of proof is used. Often the problem with this sort of analysis is that the bounds can grow very rapidly (Wilkinson [59]). This is not the case here. A certain key fact is used to obtain a constant bound on multiplicative terms that would otherwise have led to an exponential growth in the bounds obtained. Once these expressions and bounds on δR and δW are obtained, expressions and bounds for $(\delta P, \delta Q)$ and $(\delta U, \delta V)$ are produced.

The primary result of this chapter is that the algorithm is weakly stable over the general class of power series. Unlike other algorithms, no restrictions need to be placed on the class of power series for this result to be valid. Moreover, except in certain exceptional cases, the algorithm is a fast $O(n^2)$ algorithm. In the very worst case, the algorithm is equivalent to Gaussian elimination with partial pivoting, which is known to be stable. Before we proceed with the actual analysis of the algorithm, we first set forth the preliminaries that are needed.

5.1 Preliminaries

We will use μ to denote the unit-roundoff of the floating point computations being done. We also assume that the degrees of all polynomials and the orders of all matrices, in what follows, are bounded by some M, where $M\mu \leq 0.01$; the restriction $M\mu \leq 0.01$ comes from Forsythe and Moler [22]. After Wilkinson [60], we denote a floating point operation by $fl(\cdot)$. The norm that we use is the infinity norm.

Lemma 5.1 *If V is an n-vector, then the infinity norm of V can be obtained exactly using floating point arithmetic; i.e.,*

$$fl(\| V \|_\infty) = \| V \|_\infty.$$

A standard result in floating point error analysis describes the error that occurs when a number of values are added.

Lemma 5.2 *If $n\mu \leq 0.01$, then*

$$fl(\textstyle\sum_{k=1}^{n} u_k v_k) = \displaystyle\sum_{k=1}^{n} u_k v_k (1 + \gamma_k),$$

where $|\gamma_k| \leq 1.01 n\mu$.

Proof A proof of this lemma may be found in Forsythe and Moler [22, page 93]. ∎

In our analysis, we almost always use polynomials; the following definition gives a polynomial norm that is defined in terms of vector norms. We also use power series, so the infinity norm for a power series is likewise defined.

Definition 5.3 *If* $U(z) = \sum_{i=0}^{n} u_i z^i$, *then define the p-norm for* $U(z)$ *to be*

$$\| U \|_p = \| (u_0, u_1, \cdots, u_n)^T \|_p.$$

If $A(z) = \sum_{i=0}^{\infty} a_i z^i$, *then define the infinity-norm for* $A(z)$ *to be*

$$\| A \|_\infty = sup\{|a_i| : 0 \leq i < \infty\}.$$

In the rest of this section, we will switch between polynomials and vectors without comment, using the same notation for both; the context should indicate which we are using at any one time. In the later sections, we will almost exclusively use polynomials; where we use vectors, it shall be made explicit.

Lemma 5.4 *Let* $A(z)$ *be a power series and let* $V(z)$ *be a polynomial of degree* n. *Then*

$$\| (AV) \pmod{z^{m+1}} \|_\infty \leq (min\{m, n\} + 1) \| A \|_\infty \| V \|_\infty.$$

Proof We can express the coefficients of the product of $(AV) \pmod{z^{m+1}}$ as

$$\begin{pmatrix} (AV)_0 \\ (AV)_1 \\ \vdots \\ (AV)_m \end{pmatrix} \equiv \begin{pmatrix} 0 & & a_0 \\ & \iddots & \\ a_0 & & \vdots \\ \vdots & & \\ a_{m-n} & \cdots & a_m \end{pmatrix} \begin{pmatrix} v_n \\ \vdots \\ v_0 \end{pmatrix},$$

so

$$\begin{aligned} \| (AV) \pmod{z^{m+1}} \|_\infty &= \left\| \begin{pmatrix} 0 & & a_0 \\ & \iddots & \\ a_0 & & \vdots \\ \vdots & & \\ a_{m-n} & \cdots & a_m \end{pmatrix} \begin{pmatrix} v_n \\ \vdots \\ v_0 \end{pmatrix} \right\|_\infty \\ &\leq \left\| \begin{pmatrix} 0 & & a_0 \\ & \iddots & \\ a_0 & & \vdots \\ \vdots & & \\ a_{m-n} & \cdots & a_m \end{pmatrix} \right\|_\infty \| V \|_\infty \\ &\leq (\min\{m,n\}+1) \| A \|_\infty \| V \|_\infty \end{aligned}$$

∎

Lemma 5.5 *Let $A(z)$ be a power series and let $V(z)$ be a polynomial of degree n. Then*

$$fl((AV) \pmod{z^{m+1}}) = (AV) \pmod{z^{m+1}} + \Psi, \text{ where}$$

$$\| \Psi \|_\infty \leq 1.01(min\{m,n\}+1)^2 \| A \|_\infty \| V \|_\infty \mu.$$

Proof If we let $M = min\{m,n\}$, we have, using Lemma 5.2,

$$\begin{aligned}
(fl(AV))_i &= fl(\textstyle\sum_{j=0}^{M} a_j v_{i-j}) \\
&= \sum_{j=0}^{M} a_j v_{i-j}(1+\gamma_{ij}),\ |\gamma_{ij}| \leq 1.01(M+1)\mu \\
&= \sum_{j=0}^{M} a_j v_{i-j} + \sum_{j=0}^{M} a_j v_{i-j}\gamma_{ij}.
\end{aligned}$$

So we have,

$$\Psi = \left(\sum_{j=0}^{M} a_j v_{i-j}\gamma_{ij} \right)_{i=0,\ldots,m}.$$

From this last expression, it is easily seen that

$$\| \Psi \|_\infty \leq 1.01(M+1)^2 \| A \|_\infty \| V \|_\infty \mu.$$

∎

Lemma 5.6 *Let U and V be polynomials of degree n. Then*

$$fl(U+V) = U + V + \Psi, \text{ where}$$

$$\| \Psi \|_\infty \leq \| U+V \|_\infty \mu.$$

Proof

$$fl(U+V) = (I + diag(\varepsilon_i))(U+V),\ |\varepsilon_i| \leq \mu,\ i = 0, ..., n.$$

So

$$\Psi = diag(\varepsilon_i)(U+V)$$

and

$$\| \Psi \|_\infty \leq \| diag(\varepsilon_i) \|_\infty \| U+V \|_\infty \leq \| U+V \|_\infty \mu.$$

∎

This completes the preliminary material that we need. For the remainder of this work, we use the infinity norm; for this reason, we simply use $\| \cdot \|$ instead of the more explicit $\| \cdot \|_\infty$.

5.2 Round-Off Analysis: The Individual Steps

Now that we have presented the necessary preliminaries, we return to the analysis of the algorithm. For the rest of this chapter and those following, the previously defined quantities $\mathcal{P}^{(k)}$, $\hat{\mathcal{P}}^{(k)}$, $\mathcal{R}^{(k)}$, etcetera, are now taken to be the computed quantities that result from the use of floating point operations in NPADE. Where an exact quantity is used, it shall be explicitly indicated.

We will first examine what happens in one iteration of the algorithm. At the start of an iteration, we have just computed $\mathcal{P}^{(k)}$. At this stage, we have

$$\mathcal{P}^{(k)}\mathcal{A} = \mathcal{D}^{(k)} + O(z^{2n_k}), \tag{5.1}$$

where

$$\mathcal{A} = \begin{pmatrix} -1 \\ A \end{pmatrix} \text{ and } \mathcal{D}^{(k)} = \begin{pmatrix} \delta R^{(k)} \\ \delta W^{(k)} \end{pmatrix}.$$

$\delta R^{(k)}$ and $\delta W^{(k)}$ are polynomials of degree $2n_k - 1$, where the constant and linear terms in $\delta R^{(k)}$ are zero. They are the error in the order condition for the currently computed Padé approximant pair. Also, since we never need to calculate $\mathcal{D}^{(k)}$ explicitly, we are using the exact relation

$$\mathcal{D}^{(k)} = (\mathcal{P}^{(k)}\mathcal{A}) \pmod{z^{2n_k}}.$$

To obtain $\mathcal{P}^{(k+1)}$, the algorithm produces $\hat{\mathcal{P}}^{(k)}$ which is the computed solution to the Sylvester system based upon the residuals. $\mathcal{P}^{(k+1)}$ is obtained by performing the floating point multiplication of $\mathcal{P}^{(k)}$ by $\hat{\mathcal{P}}^{(k)}$. Then

$$\mathcal{P}^{(k+1)}\mathcal{A} = \mathcal{D}^{(k+1)} + O(z^{2n_{k+1}}).$$

From the analysis of one iteration, we obtain an expression for $\mathcal{D}^{(k+1)}$ in terms of $\mathcal{D}^{(k)}$ and the local errors that are introduced during this iteration. Once this recurrence is obtained, it is iteratively applied to give an expression for $\mathcal{D}^{(k+1)}$ in terms of the errors introduced by each iterative step of NPADE.

This is then used to analyse how close the computed solutions are to the true solutions.

To analyse one iteration, we examine each of the three steps that make up an iteration. They are: 1) calculation of the residual $\mathcal{R}^{(k)}$; 2) solving the associated Sylvester system to obtain $\hat{\mathcal{P}}^{(k)}$; and 3) using $\hat{\mathcal{P}}^{(k)}$ and $\mathcal{P}^{(k)}$ to generate $\mathcal{P}^{(k+1)}$ through matrix multiplication. In what follows, we are interested in the effect that the use of floating point operations has on the values computed. We now proceed to obtain expressions and bounds for the local errors introduced in each of the three steps.

Most of the quantities that we use in the analysis are either two by two matrices (e.g., $\mathcal{P}^{(k)}$) or two-vectors (e.g., $\mathcal{D}^{(k)}$) whose entries are polynomials. As a notational convenience, we use $|\cdot|$ to denote the componentwise application of the infinity norm. For example,

$$|\hat{\mathcal{P}}^{(k)}| = \begin{pmatrix} \| \hat{P}^{(k)} \| & \| \hat{Q}^{(k)} \| \\ \| \hat{U}^{(k)} \| & \| \hat{V}^{(k)} \| \end{pmatrix}.$$

We also rely on certain normalizations that have been applied to various quantities, namely:

$$\| P^{(k)} \| + \| Q^{(k)} \| = 1, \tag{5.2}$$

$$\| U^{(k)} \| + \| V^{(k)} \| = 1, \tag{5.3}$$

$$\| A \| = 1. \tag{5.4}$$

A comment must be made about the bounds that are obtained. The

constants in the expressions become somewhat large. Most of these constants are due to the use of the infinity norm and the inequality given by Lemma 5.4 (e.g., if U and V are polynomials of degree n, then $\| UV \| \leq (n+1)\| U \|\| V \|$). While this inequality can be strict, in practice, it is not usually so. Our choice of the infinity norm is not accidental. It plays a critical role in obtaining bounds on quantities that describe the propagation of errors from one step to the next. It also enables the algorithm to use a fixed ϵ for acceptance of a solution instead of a dynamic one.

5.2.1 Residual Calculation

During this phase, the algorithm computes the first $2s_k$ terms of $\mathcal{R}^{(k)}$. Since $B \equiv -1$, we have

$$z^{2n_k}\mathcal{R}^{(k)} = \begin{pmatrix} fl((AQ^{(k)}) \pmod{z^{2n_k+2s_k}}) \\ fl((AV^{(k)}) \pmod{z^{2n_k+s_k}}) \end{pmatrix} - \begin{pmatrix} fl((AQ^{(k)}) \pmod{z^{2n_k}}) \\ fl((AV^{(k)}) \pmod{z^{2n_k}}) \end{pmatrix},$$

so that $\mathcal{R}^{(k)}$ satisfies

$$\mathcal{P}^{(k)}\mathcal{A} \pmod{z^{2n_k+2s_k}} = \mathcal{D}^{(k)} + z^{2n_k}(\mathcal{R}^{(k)} - \mathcal{C}_1^{(k)}),$$

where $\mathcal{C}_1^{(k)}$ is the error in the computed residual introduced by the floating point operations.

Lemma 5.7 *The computed $\mathcal{R}^{(k)}$ satisfies*

$$z^{2n_k}\mathcal{R}^{(k)} = \mathcal{P}^{(k)}\mathcal{A} - \mathcal{D}^{(k)} + z^{2n_k}\mathcal{C}_1^{(k)} \tag{5.5}$$

where

$$|C_1^{(k)}| \leq 1.01(n_k+1)^2\mu \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \qquad (5.6)$$

Proof This follows immediately from Lemma 5.5 and the normalizations (5.2), (5.3), and (5.4). ∎

5.2.2 Solution of the Sylvester System

In this phase, a Sylvester system, with the matrix

$$\hat{S}_k = \begin{pmatrix} 0 & 0 & 0 & r_0 & 0 & 0 \\ w_0 & & 0 & \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots & \vdots & & r_0 \\ \vdots & & w_0 & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & & \vdots \\ w_{2s_k-2} & \cdots & w_{s_k-1} & r_{2s_k-1} & \cdots & r_{s_k} \end{pmatrix}, \qquad (5.7)$$

is solved to obtain $\hat{\mathcal{P}}^{(k)}$. This computed $\hat{\mathcal{P}}^{(k)}$ is not the exact solution of the equivalent problem $\hat{\mathcal{P}}^{(k)}\mathcal{R}^{(k)} = O(z^{2s_k})$. However, we can describe the errors that are introduced during the solution of the system.

Lemma 5.8

$$\| \hat{S}_k \| \leq 2s_k(n_k+1)(1+1.01(n_k+1)\mu)$$

Proof Clearly, $\| \hat{S}_k \| \leq s_k(\| W^{(k)} \| + \| R^{(k)} \|)$. Now, using Lemmas 5.4 and 5.5,

$$\begin{aligned} \| W^{(k)} \| &\leq \| AV^{(k)} \| \\ &\leq (n_k + 1)(1 + 1.01(n_k + 1)\mu)\| A \|\| V^{(k)} \| \end{aligned}$$

and

$$\begin{aligned} \| R^{(k)} \| &\leq \| AQ^{(k)} \| \\ &\leq n_k(1 + 1.01n_k\mu)\| A \|\| Q^{(k)} \|. \end{aligned}$$

By applying (5.2), (5.3), and (5.4), the result follows.[1] ∎

To obtain

$$\hat{\mathcal{P}}^{(k)} = \begin{pmatrix} \hat{P}^{(k)} & \hat{Q}^{(k)} \\ \hat{U}^{(k)} & \hat{V}^{(k)} \end{pmatrix},$$

[1] A comment must be made about this bound. Experiments have shown that in practice, for random power series this norm is near unity and for power series defined by convergent sequences, the norm is much less than one. These are examples that illustrate the coarseness of some of these bounds.

the algorithm uses Gaussian elimination with partial pivoting. Letting

$$\vec{V} = \begin{pmatrix} \hat{v}_1 \\ \vdots \\ \hat{v}_{s_k} \\ \hat{u}_0 \\ \vdots \\ \hat{u}_{s_k-1} \end{pmatrix}, \vec{W} = \begin{pmatrix} w_0 \\ \vdots \\ w_{2s_k-1} \end{pmatrix}, \text{ and } \vec{Q} = \begin{pmatrix} \hat{q}_2 \\ \vdots \\ \hat{q}_{s_k+1} \\ 0 \\ \hat{p}_2 \\ \vdots \\ \hat{p}_{s_k} \end{pmatrix},$$

$\hat{\mathcal{P}}^{(k)}$ is obtained by solving

$$\hat{S}_k \vec{Q} = \hat{r}_0 e_{2s_k} \tag{5.8}$$

and

$$\hat{S}_k \vec{V} = -\hat{v}_0 \vec{W}, \tag{5.9}$$

where $-\hat{v}_0$ and $\hat{r}_0$ are parameters (keeping in mind that $\hat{p}_0 = \hat{p}_1 = \hat{q}_0 = \hat{q}_1 = 0$)[2]. If $\hat{S}_k$ is nonsingular, then $-\hat{v}_0$ and $\hat{r}_0$ are set to 1 and (5.8) and (5.9) are solved. If $\hat{S}_k$ is singular, then a nontrivial solution is obtained for when $-\hat{v}_0$ and $\hat{r}_0$ are 0. The solution that results from a singular system is only used if this is the last system solved by the algorithm (for in this case, $|r_0^{k+1} v_0^{k+1}| = 0$, so that for intermediate systems the solution is rejected). In this case, the algorithm prints the Padé form, but the analysis we give is not applicable to this unusual case.

[2]The case $s_k = 1$ is covered as well, as then $\hat{P}^{(k)} = 0$ and $\hat{Q}^{(k)} = z^2/w_0$.

Lemma 5.9 *If $\hat{S}_k$ is nonsingular and $\hat{\mathcal{P}}^{(k)}$ is obtained by solving (5.8) and (5.9) using Gaussian elimination with partial pivoting, then*

$$\hat{\mathcal{P}}^{(k)}\mathcal{R}^{(k)} = \mathcal{C}_2^{(k)} + O(z^{2s_k}), \tag{5.10}$$

where

$$|\mathcal{C}_2^{(k)}| \leq (128s_k^4\rho_k(n_k+1)\mu + O(\mu^2)) \begin{pmatrix} max\{\| \hat{P}^{(k)} \|, \| \hat{Q}^{(k)} \|\} \\ max\{\| \hat{U}^{(k)} \|, \| \hat{V}^{(k)} \|\} \end{pmatrix} \tag{5.11}$$

and ρ_k is a constant of order unity in practice.

Proof When (5.8) and (5.9) are solved using Gaussian elimination with partial pivoting, the solutions obtained are exact solutions of

$$(\hat{S}_k + E_Q)\vec{Q} = \hat{r}_0 e_{2s_k} \tag{5.12}$$

and

$$(\hat{S}_k + E_V)\vec{V} = -\hat{v}_0\vec{W}, \tag{5.13}$$

where

$$\| E_{(\cdot)} \| \leq 8(2s_k)^3\rho_k\| \hat{S}_k \|\mu + O(\mu^2)$$

and ρ_k is the growth factor associated with the LU-decomposition of $\hat{S}_k$ (Golub and van Loan [27, page 67]). With partial pivoting, ρ_k is of order

unity in practice[3]. Now from (5.12) and (5.13), we have

$$\hat{S}_k\vec{Q} - \hat{r}_0 e_{2s_k} = -E_Q\vec{Q} \tag{5.14}$$

and

$$\hat{S}_k\vec{V} + \hat{v}_0\vec{W} = -E_V\vec{V}, \tag{5.15}$$

where

$$\begin{pmatrix} \| E_Q\vec{Q} \| \\ \| E_V\vec{V} \| \end{pmatrix} \leq (64 s_k^3 \rho_k \| \hat{S}_k \| \mu + O(\mu^2)) \begin{pmatrix} max\{\| \hat{P}^{(k)} \|, \| \hat{Q}^{(k)} \|\} \\ max\{\| \hat{U}^{(k)} \|, \| \hat{V}^{(k)} \|\} \end{pmatrix}.$$

But using the bound on $\| \hat{S}_k \|$ from Lemma 5.8 and absorbing the terms that are nonlinear in μ into the $O(\mu^2)$ term, the expression on the right-hand-side of (5.11) is obtained. Now, since a $\hat{\mathcal{P}}^{(k)}$ which satisfies $\hat{\mathcal{P}}^{(k)}\mathcal{R}^{(k)} = O(z^{2s_k})$ can be found by solving (5.8) and (5.9) exactly, it is easy to see the correspondence between (5.10) and the linear systems given by (5.12) and (5.13). From this, we have

$$\mathcal{C}_2^{(k)} = \begin{pmatrix} -E_Q\vec{Q} \\ -E_V\vec{V} \end{pmatrix}$$

and the result follows. ∎

[3]Examples can be constructed where the growth factor ρ grows exponentially if partial pivoting is used, but in practice ρ is usually comparable to the modest growth that results when complete pivoting is used (which is approximately 10 in practice)(Golub and van Loan [27, page 69]). Further discussion and new results regarding the growth factor ρ can be found in Higham and Higham [31] and Trefethen and Schreiber [56].

5.2.3 Applying the Recurrence Relation

With $\hat{\mathcal{P}}^{(k)}$ computed, the next node can be obtained by application of Lemma 2.10. However, since the arithmetic is not exact, we have the following.

Lemma 5.10 *If* $\mathcal{P}^{(k+1)} = fl(\hat{\mathcal{P}}^{(k)}\mathcal{P}^{(k)})$, *then*

$$\mathcal{P}^{(k+1)} = \hat{\mathcal{P}}^{(k)}\mathcal{P}^{(k)} + \mathcal{C}_3^{(k)}, \tag{5.16}$$

where

$$|\mathcal{C}_3^{(k)}| \leq (2.02(s_k+1)^2\mu + O(\mu^2))|\hat{\mathcal{P}}^{(k)}||\mathcal{P}^{(k)}|. \tag{5.17}$$

Proof We prove this result for the $(1,1)$ entry in (5.16), namely, $P^{(k+1)}$; the other entries follow by the same argument, thereby yielding (5.17).

$$\begin{aligned} P^{(k+1)} &= fl(fl(U^{(k)}\hat{Q}^{(k)}) + fl(P^{(k)}\hat{P}^{(k)})) \\ &= fl((U^{(k)}\hat{Q}^{(k)} + \zeta) + (P^{(k)}\hat{P}^{(k)} + \eta)) \\ &= U^{(k)}\hat{Q}^{(k)} + P^{(k)}\hat{P}^{(k)} + \zeta + \eta + \xi \end{aligned}$$

where, by Lemma 5.5,

$$\begin{aligned} \| \zeta \| &\leq 1.01 s_k^2 \| U^{(k)} \| \| \hat{Q}^{(k)} \| \mu \\ \| \eta \| &\leq 1.01 (s_k - 1)^2 \| P^{(k)} \| \| \hat{P}^{(k)} \| \mu \end{aligned}$$

and, by Lemma 5.6,

$$\begin{aligned} \| \xi \| &\leq \| U^{(k)}\hat{Q}^{(k)} + P^{(k)}\hat{P}^{(k)} + \zeta + \eta \| \mu \\ &\leq (\| U^{(k)}\hat{Q}^{(k)} \| + \| P^{(k)}\hat{P}^{(k)} \|)\mu + (\| \zeta \| + \| \eta \|)\mu \\ &\leq s_k(\| U^{(k)} \| \| \hat{Q}^{(k)} \| + \| P^{(k)} \| \| \hat{P}^{(k)} \|)\mu + (\| \zeta \| + \| \eta \|)\mu. \end{aligned}$$

Thus, the error, $(\zeta + \eta + \xi)$, is bounded by

$$\begin{aligned}
\| \zeta + \eta + \xi \| &\leq s_k(\| U^{(k)} \| \| \hat{Q}^{(k)} \| + \| P^{(k)} \| \| \hat{P}^{(k)} \|)\mu \\
&\quad + (\| \zeta \| + \| \eta \|)(1 + \mu) \\
&\leq s_k(\| U^{(k)} \| \| \hat{Q}^{(k)} \| + \| P^{(k)} \| \| \hat{P}^{(k)} \|)\mu \\
&\quad + (1.01 s_k^2(\| U^{(k)} \| \| \hat{Q}^{(k)} \| + \| P^{(k)} \hat{P}^{(k)} \|)\mu(1 + \mu)) \\
&= (\| U^{(k)} \| \| \hat{Q}^{(k)} \| + \| P^{(k)} \| \| \hat{P}^{(k)} \|)(s_k\mu + 1.01 s_k^2 \mu(1 + \mu)) \\
&= (\| \hat{P}^{(k)} \| \| \hat{Q}^{(k)} \|) \cdot (\| P^{(k)} \| \| U^{(k)} \|)^T \\
&\quad \cdot (s_k\mu(1 + 1.01 s_k) + 1.01 s_k^2 \mu^2) \\
&\leq (\| \hat{P}^{(k)} \| \| \hat{Q}^{(k)} \|) \cdot (\| P^{(k)} \| \| U^{(k)} \|)^T (2.02 s_k^2 \mu + O(\mu^2)).
\end{aligned}$$

∎

At this point, $\mathcal{P}^{(k+1)}$ and $\hat{\mathcal{P}}^{(k)}$ are scaled so that

$$\| P^{(k+1)} \| + \| Q^{(k+1)} \| = 1$$

and

$$\| U^{(k+1)} \| + \| V^{(k+1)} \| = 1.$$

We assume that this introduces no additional error. In our analysis, we only actually require 1 to be an upper bound for the above quantities. We could modify the normalizations and rescaling so that they could be done exactly.

However, since we are dealing with the infinity norm and the errors that result from this normalization are small, we have made this assumption for the sake of simplicity. Now, this scaling has no effect on any of the prior results as in the solution of (5.8) and (5.9), $\hat{r}_0$ and $\hat{v}_0$ are just parameters.

We now have expressions defining the errors introduced by each phase of one iterative step of the algorithm. In the next section, we combine these to obtain the local error introduced by one step and then use this to obtain the cumulative error in the Padé approximant order condition.

5.3 Cumulative Residual Error

For purposes of analysis, it is desirable to represent the local error in a slightly different form. Let

$$\mathcal{L}_1^{(k)} = -z^{2n_k}\hat{\mathcal{P}}^{(k)}\mathcal{C}_1^{(k)} \pmod{z^{2n_{k+1}}}, \tag{5.18}$$

$$\mathcal{L}_2^{(k)} = z^{2n_k}\mathcal{C}_2^{(k)} \pmod{z^{2n_{k+1}}}, \text{ and} \tag{5.19}$$

$$\mathcal{L}_3^{(k)} = \mathcal{C}_3^{(k)}\mathcal{A} \pmod{z^{2n_{k+1}}}. \tag{5.20}$$

$\mathcal{L}_1^{(k)}$ comes from the floating point computation of the residual $\mathcal{R}^{(k)}$ (see (5.5)), $\mathcal{L}_2^{(k)}$ comes from solving the residual Sylvester system (see (5.10)), and $\mathcal{L}_3^{(k)}$ comes from advancing the solution through multiplication by $\hat{\mathcal{P}}^{(k)}$ (see (5.16)). The sum of these three quantities, (5.18), (5.19), and (5.20), gives the total local error. More specifically,

Lemma 5.11 *Let the error $\mathcal{D}^{(k)}$ satisfy*

$$\mathcal{P}^{(k)}\mathcal{A} = \mathcal{D}^{(k)} + O(z^{2n_k}) \tag{5.21}$$

at the start of the k^{th} iteration. Then the residual error $\mathcal{D}^{(k+1)}$ at the start of the $(k+1)^{st}$ iteration satisfies

$$\mathcal{D}^{(k+1)} = \hat{\mathcal{P}}^{(k)}\mathcal{D}^{(k)} + \mathcal{L}^{(k)}, \tag{5.22}$$

where $\mathcal{L}^{(k)} = \mathcal{L}_1^{(k)} + \mathcal{L}_2^{(k)} + \mathcal{L}_3^{(k)}$.

Proof We have, using (5.16), (5.20), (5.5), (5.18), (5.10), and (5.19),

$$\begin{aligned}
\mathcal{D}^{(k+1)} &= (\mathcal{P}^{(k+1)}\mathcal{A}) \pmod{z^{2n_{k+1}}} \\
&= ((\hat{\mathcal{P}}^{(k)}\mathcal{P}^{(k)} + \mathcal{C}_3^{(k)})\mathcal{A}) \pmod{z^{2n_{k+1}}} \\
&= (\hat{\mathcal{P}}^{(k)}(\mathcal{P}^{(k)}\mathcal{A})) \pmod{z^{2n_{k+1}}} + \mathcal{L}_3^{(k)} \\
&= (\hat{\mathcal{P}}^{(k)}(z^{2n_k}\mathcal{R}^{(k)} + \mathcal{D}^{(k)} - z^{2n_k}\mathcal{C}_1^{(k)})) \pmod{z^{2n_{k+1}}} \\
&\quad + \mathcal{L}_3^{(k)} \\
&= (z^{2n_k}(\hat{\mathcal{P}}^{(k)}\mathcal{R}^{(k)}) + \hat{\mathcal{P}}^{(k)}\mathcal{D}^{(k)}) \pmod{z^{2n_{k+1}}} \\
&\quad + \mathcal{L}_1^{(k)} + \mathcal{L}_3^{(k)} \\
&= (z^{2n_k}(\mathcal{C}_2^{(k)} + O(z^{2s_k})) \pmod{z^{2n_{k+1}}}) + \hat{\mathcal{P}}^{(k)}\mathcal{D}^{(k)} + \mathcal{L}_1^{(k)} + \mathcal{L}_3^{(k)} \\
&= \hat{\mathcal{P}}^{(k)}\mathcal{D}^{(k)} + \mathcal{L}_1^{(k)} + \mathcal{L}_2^{(k)} + \mathcal{L}_3^{(k)}.
\end{aligned}$$

∎

From (5.22), we see that $\hat{\mathcal{P}}^{(k)}$ is the propagation matrix for the error. Its size will determine how the error introduced in one step affects later steps. We will see shortly that its size is determined by the normalizations (5.2) and (5.3) and $|r_0^{(k+1)} v_0^{(k+1)}|$. The next theorem details the propagation of the local error to subsequent steps.

Theorem 5.12

$$\mathcal{D}^{(k)} = \sum_{i=0}^{k-1} \mathcal{G}_{i+1}^{(k-1)} \mathcal{L}^{(i)}, \tag{5.23}$$

where

$$\begin{aligned} \mathcal{G}_i^{(j)} &= \hat{\mathcal{P}}^{(j)} \hat{\mathcal{P}}^{(j-1)} \ldots \hat{\mathcal{P}}^{(i)}, \; 0 \le i \le j, \; \textit{and} \\ \mathcal{G}_{j+1}^{(j)} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

Proof This follows from Lemma 5.11 by induction. ∎

From (5.23), we see that $\mathcal{G}_i^{(k)}$ describes the way that an error introduced at the $(i-1)^{th}$ stage in the algorithm is propagated through to the kth stage.

We now proceed to obtain a bound on $\mathcal{D}^{(k)}$. While (5.23) describes the total error in a natural fashion, to obtain the bound we will want to consider the local errors comprising (5.23) separately. Let

$$\mathcal{E}_i^{(k)} = \sum_{l=0}^{k} \mathcal{G}_{l+1}^{(k)} \mathcal{L}_i^{(l)}, \; i = 1, 2, 3. \tag{5.24}$$

Since, $\mathcal{L}^{(k)} = \mathcal{L}_1^{(k)} + \mathcal{L}_2^{(k)} + \mathcal{L}_3^{(k)}$, then clearly (5.23) is equivalent to

$$\mathcal{D}^{(k)} = \mathcal{E}_1^{(k-1)} + \mathcal{E}_2^{(k-1)} + \mathcal{E}_3^{(k-1)}.$$

From (5.18), (5.19), (5.20), and (5.24), we have explicitly

$$\mathcal{E}_1^{(k)} = \sum_{l=0}^{k} -z^{2n_l}\mathcal{G}_{l+1}^{(k)}(\hat{\mathcal{P}}^{(l)}\mathcal{C}_1^{(l)} \pmod{z^{2s_l}}), \text{ where } \mathcal{C}_1^{(0)} = 0, \quad (5.25)$$

$$\mathcal{E}_2^{(k)} = \sum_{l=0}^{k} z^{2n_l}\mathcal{G}_{l+1}^{(k)}(\mathcal{C}_2^{(l)} \pmod{z^{2s_l}}), \text{ and} \quad (5.26)$$

$$\mathcal{E}_3^{(k)} = \sum_{l=0}^{k} \mathcal{G}_{l+1}^{(k)}((\mathcal{C}_3^{(l)}\mathcal{A}) \pmod{z^{2n_{l+1}}}), \text{ where } \mathcal{C}_3^{(0)} = 0. \quad (5.27)$$

In (5.25) and (5.27), we have used some results that are implicit from the statement of the algorithm, namely, $\mathcal{C}_1^{(0)} = \mathcal{C}_3^{(0)} = 0$. To see this, we briefly consider the first pass through the While loop in the algorithm. In the first pass,

$$\mathcal{P}^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

so that $\mathcal{R}^{(0)} = \mathcal{A} \pmod{z^{2s_0}}$ exactly, so that there is no error in computing this first residual; i.e., $\mathcal{C}_1^{(0)} = 0$. Once the algorithm has computed $\hat{\mathcal{P}}^{(0)}$, it then computes $fl(\hat{\mathcal{P}}^{(0)}\mathcal{P}^{(0)})$. Again, since $\mathcal{P}^{(0)}$ is the identity matrix, we have no error in this step, so $\mathcal{C}_3^{(0)} = 0$.

Returning to our primary goal of obtaining a bound for the total error, we observe that the key expression to bound is $\mathcal{G}_i^{(k)}$. A concern here is that the $\hat{\mathcal{P}}^{(j)}$'s making up $\mathcal{G}_i^{(k)}$ will cause exponential growth with k in $|\mathcal{G}_i^{(k)}|$.

However, we will show that this is not the case. A consequence of this, then, will be that the error as described by (5.23), will be the sum of the errors introduced in each step. To obtain this bound on $\mathcal{G}_i^{(k)}$, we use a relation from Chapter 4, (4.4), which we restate for $m_k = n_k - 1$, namely,

$$Q^{(k)}U^{(k)} - V^{(k)}P^{(k)} = \vartheta^{(k)}(z) + r_0^k v_0^k z^{2n_k}, \tag{5.28}$$

where $\vartheta^{(k)}(z) = (V^{(k)}\delta R^{(k)} - Q^{(k)}\delta W^{(k)}) \pmod{z^{2n_k+1}}$. Now, since

$$\mathcal{D}^{(k)} = \begin{pmatrix} \delta R^{(k)} \\ \delta W^{(k)} \end{pmatrix},$$

we see that $\vartheta^{(k)}(z)$ depends on $\mathcal{D}^{(k)}$.

If we let

$$\mathcal{P}_I^{(k)} = \begin{pmatrix} V^{(k)} & -Q^{(k)} \\ -U^{(k)} & P^{(k)} \end{pmatrix}$$

then

$$\mathcal{P}^{(k)}\mathcal{P}_I^{(k)} = -\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}(r_0^k v_0^k z^{2n_k} + \vartheta^{(k)}(z)). \tag{5.29}$$

With this, we are now able to prove the following lemma.

Lemma 5.13 *Assuming that* $r_0^{(k)}$ *and* $v_0^{(k)}$ *are large and* $\| \delta R^{(k)} \|$ *and* $\| \delta W^{(k)} \|$ *are small so that*

$$22(n_k+1)^2(\| \delta R^{(k)} \| + \| \delta W^{(k)} \| + 5(n_k+1)\mu) \leq |r_0^{(k)}v_0^{(k)}|, \tag{5.30}$$

then

$$|\mathcal{G}_k^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \leq \frac{4(n_k+1)}{|r_0^{(k)}v_0^{(k)}|} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{5.31}$$

Proof From (5.16), we have that

$$|\mathcal{P}^{(k+1)}\mathcal{P}_I^{(k)}|\begin{pmatrix}1\\1\end{pmatrix} = |\hat{\mathcal{P}}^{(k)}\mathcal{P}^{(k)}\mathcal{P}_I^{(k)} + \mathcal{C}_3^{(k)}\mathcal{P}_I^{(k)}|\begin{pmatrix}1\\1\end{pmatrix} \tag{5.32}$$

Now, using (5.2) and (5.3),

$$|\mathcal{P}^{(k+1)}||\mathcal{P}_I^{(k)}|\begin{pmatrix}1\\1\end{pmatrix} \le |\mathcal{P}^{(k+1)}|\begin{pmatrix}2\\2\end{pmatrix} = \begin{pmatrix}2\\2\end{pmatrix}, \tag{5.33}$$

so that the left-hand-side of (5.32) is bounded by

$$\begin{aligned}|\mathcal{P}^{(k+1)}\mathcal{P}_I^{(k)}|\begin{pmatrix}1\\1\end{pmatrix} &\le (n_k+1)|\mathcal{P}^{(k+1)}||\mathcal{P}_I^{(k)}|\begin{pmatrix}1\\1\end{pmatrix}\\ &\le 2(n_k+1)|\mathcal{P}^{(k+1)}|\begin{pmatrix}1\\1\end{pmatrix}\\ &\le 2(n_k+1)\begin{pmatrix}1\\1\end{pmatrix}. \end{aligned} \tag{5.34}$$

On the other hand, using (5.29), Lemma (5.5), and (5.17), the right-hand-side of (5.32) is bounded below by:

$$\begin{aligned}&|\hat{\mathcal{P}}^{(k)}\mathcal{P}^{(k)}\mathcal{P}_I^{(k)} + \mathcal{C}_3^{(k)}\mathcal{P}_I^{(k)}|\begin{pmatrix}1\\1\end{pmatrix}\\ &\ge |\hat{\mathcal{P}}^{(k)}(|r_0^{(k)}v_0^{(k)}|z^{2n_k})|\begin{pmatrix}1\\1\end{pmatrix} - |\hat{\mathcal{P}}^{(k)}\vartheta^{(k)}(z)|\begin{pmatrix}1\\1\end{pmatrix}\end{aligned}$$

$$
\begin{aligned}
& - (n_k+1)|\mathcal{C}_3^{(k)}||\mathcal{P}_I^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
\geq\ & |r_0^{(k)} v_0^{(k)}||\hat{\mathcal{P}}^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} - (s_k+1)\| \vartheta^{(k)}(z) \| \cdot |\hat{\mathcal{P}}^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad (5.35) \\
& - (n_k+1)[2.02(s_k+1)^2\mu + O(\mu^2)]|\hat{\mathcal{P}}^{(k)}||\mathcal{P}^{(k)}||\mathcal{P}_I^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix}.
\end{aligned}
$$

Now, using the same argument as used in (5.33), (5.35) gives us

$$
\begin{aligned}
& |\hat{\mathcal{P}}^{(k)}\mathcal{P}^{(k)}\mathcal{P}_I^{(k)} + \mathcal{C}_3^{(k)}\mathcal{P}_I^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
\geq\ & |r_0^{(k)} v_0^{(k)}||\hat{\mathcal{P}}^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} - (s_k+1)(n_k+1)(\| \delta R^{(k)} \| + \| \delta W^{(k)} \|)|\hat{\mathcal{P}}^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
& - (4.04(n_k+1)(s_k+1)^2\mu + O(\mu^2))|\hat{\mathcal{P}}^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
=\ & \left(|r_0^{(k)} v_0^{(k)}| - [(s_k+1)(n_k+1)(\| \delta R^{(k)} \| + \| \delta W^{(k)} \| + 4.04(s_k+1)\mu + O(\mu^2)\}]\right) \\
& |\hat{\mathcal{P}}^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
\geq\ & |\hat{\mathcal{P}}^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \frac{|r_0^{(k)} v_0^{(k)}|}{2}. \qquad (5.36)
\end{aligned}
$$

The last inequality follows from the application of (5.30), if we assume that $s_k \leq n_k$. But the above argument also holds, with minor modifications, if

$s_k > n_k$, so that (5.36) is valid in general[4].

Since $\mathcal{G}_k^{(k)} = \hat{\mathcal{P}}^{(k)}$, it follows from (5.34), (5.36), and (5.32), that

$$|\mathcal{G}_k^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \leq \frac{4(n_k+1)}{|r_0^{(k)} v_0^{(k)}|} \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

To simplify our notation, we shall let

$$\sigma^{(k)} = (r_0^{(k)} v_0^{(k)}). \tag{5.37}$$

Corollary 5.14 *Given that (5.30) holds,*

$$|\mathcal{C}_3^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \leq \left(\frac{8.08(s_k+1)^2(n_k+1)}{|\sigma^{(k)}|} \mu + O(\mu^2) \right) \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{5.38}$$

Proof This result is immediate from (5.17) and Lemma 5.13. ∎

These two results now enable us to obtain a bound for the general $\mathcal{G}_i^{(k)}$. This bound is significant as $\mathcal{G}_i^{(k)}$ describes the propagation of the local error in a single step to later steps. This bound that we obtain essentially says that $|\mathcal{G}_i^{(k)}|$ is independent of k. In other words, when the error is introduced, it does not grow significantly after its introduction. A consequence of this is that the errors that occur in each step are additive.

[4]The application of (5.30) only requires a constant factor of 2 (or 6 if $s_k > n_k$) instead of 22. However, the larger factor is required later in Lemma 5.20.

Theorem 5.15 *If the conditions of Lemma 5.13 hold for each $i \le k$, then*

$$|\mathcal{G}_i^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \le \frac{4(n_i+1)+O(\mu)}{|\sigma^{(i)}|} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{5.39}$$

Proof By repeated application of the recurrence relation, (5.16), we have

$$\mathcal{P}^{(k+1)} = \mathcal{G}_i^{(k)} \mathcal{P}^{(i)} + \sum_{l=i}^{k} \mathcal{G}_{l+1}^{(k)} \mathcal{C}_3^{(l)}.$$

The result is then proved by induction on i, starting with $i = k$. For $i = k$, (5.39) reduces to (5.31).

Now assume that (5.39) holds for $\mathcal{G}_k^{(k)}, \mathcal{G}_{k-1}^{(k)}, \ldots, \mathcal{G}_i^{(k)}$. From

$$\mathcal{P}^{(k+1)} = \mathcal{G}_{i-1}^{(k)} \mathcal{P}^{(i-1)} + \sum_{l=i-1}^{k} \mathcal{G}_{l+1}^{(k)} \mathcal{C}_3^{(l)},$$

it follows that

$$|\mathcal{P}^{(k+1)} \mathcal{P}_I^{(i-1)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} = |\mathcal{G}_{i-1}^{(k)} \mathcal{P}^{(i-1)} \mathcal{P}_I^{(i-1)} + \sum_{l=i-1}^{k} \mathcal{G}_{l+1}^{(k)} \mathcal{C}_3^{(l)} \mathcal{P}_I^{(i-1)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

As in the proof of Lemma 5.13, we can show that

$$|\mathcal{P}^{(k+1)} \mathcal{P}_I^{(i-1)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \le 2(n_{i-1}+1) \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

On the other hand, using

$$\mathcal{P}^{(i-1)} \mathcal{P}_I^{(i-1)} = - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} (r_0^{(i-1)} v_0^{(i-1)} z^{2n_{i-1}} + \vartheta^{(i-1)}(z)), \tag{5.40}$$

(5.2), (5.3), (5.38), $\partial(\vartheta^{(i-1)}) \leq 3n_{i-1} - 1$, and the inductive hypothesis,

$$
\begin{aligned}
&|\mathcal{G}_{i-1}^{(k)}\mathcal{P}^{(i-1)}\mathcal{P}_I^{(i-1)} + \sum_{l=i-1}^{k} \mathcal{G}_{l+1}^{(k)}\mathcal{C}_3^{(l)}\mathcal{P}_I^{(i-1)}| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} \\
&\geq |\mathcal{G}_{i-1}^{(k)}\left(-\sigma^{(i-1)}z^{2n_{i-1}} - \vartheta^{(i-1)}(z)\right)| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} - \sum_{l=i-1}^{k} |\mathcal{G}_{l+1}^{(k)}\mathcal{C}_3^{(l)}\mathcal{P}_I^{(i-1)}| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} \\
&\geq |\sigma^{(i-1)}||\mathcal{G}_{i-1}^{(k)}| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} - 3n_{i-1}|\mathcal{G}_{i-1}^{(k)}| \cdot \| \vartheta^{(i-1)}(z) \| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} \\
&\quad - \sum_{l=i-1}^{k} (n_{i-1}+1)|\mathcal{G}_{l+1}^{(k)}\mathcal{C}_3^{(l)}||\mathcal{P}_I^{(i-1)}| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} \\
&\geq |\sigma^{(i-1)}||\mathcal{G}_{i-1}^{(k)}| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} - 3n_{i-1}(n_{i-1}+1)(\| \delta R^{(i-1)} \| + \| \delta W^{(i-1)} \|)|\mathcal{G}_{i-1}^{(k)}| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} \\
&\quad - \sum_{l=i-1}^{k} (n_{i-1}+1)(n_{l+1}+1)|\mathcal{G}_{l+1}^{(k)}||\mathcal{C}_3^{(l)}||\mathcal{P}_I^{(i-1)}| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} \\
&\geq |\mathcal{G}_{i-1}^{(k)}| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} (|\sigma^{(i-1)}| - 3n_{i-1}(n_{i-1}+1)(\| \delta R^{(i-1)} \| + \| \delta W^{(i-1)} \|)) \\
&\quad - \sum_{l=i-1}^{k} 2(n_{i-1}+1)(n_{l+1}+1) \\
&\quad \cdot \left(\frac{4(n_{l+1}+1) + O(\mu)}{|\sigma^{(l+1)}|}\right)\left(\frac{8.08(s_l+1)^2(n_l+1)\mu + O(\mu^2)}{|\sigma^{(l)}|}\right) \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} \\
&\geq \frac{|\sigma^{(i-1)}|}{2}|\mathcal{G}_{i-1}^{(k)}| \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} - O(\mu) \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix},
\end{aligned}
$$

so that

$$|\mathcal{G}_{i-1}^{(k)}| \leq \frac{4(n_{i-1}+1)+O(\mu)}{|\sigma^{(i-1)}|}\begin{pmatrix}1\\1\end{pmatrix}$$

and the result follows by induction. ∎

In the above theorem, we have taken the liberty of replacing a summation involving terms linear in μ with an $O(\mu)$ expression. We could have left the summation in explicitly, but, as we shall see, this summation becomes quadratic in μ when it is used to obtain a bound on $\mathcal{D}^{(k)}$.

The conditions on $|\sigma^{(k)}| = |r_0^{(k)} v_0^{(k)}|$ and $\delta R^{(k)}$ and $\delta W^{(k)}$ require some explanation as well, but we defer this until after the next sequence of results. These results will indicate why these assumptions are reasonable.

Lemma 5.16 *If the assumptions of Theorem 5.15 hold, then*

$$\begin{aligned}|\mathcal{E}_1^{(k)}| \;\leq\; & \left(\frac{4.04(s_k+1)(n_k+1)^3}{|\sigma^{(k)}|}\right.\\ & \left.+\sum_{l=0}^{k-1}\frac{32.32 s_l(s_l+1)(n_{l+1}+1)(n_l+1)^3}{|\sigma^{(l)}\sigma^{(l+1)}|}\right)\mu\begin{pmatrix}1\\1\end{pmatrix}\\ & +O(\mu^2)\begin{pmatrix}1\\1\end{pmatrix}\end{aligned} \tag{5.41}$$

$$
\begin{aligned}
|\mathcal{E}_2^{(k)}| \;\leq\; & \left(\frac{512 s_k^4 \rho_k (n_k+1)^2 \mu}{|\sigma^{(k)}|}\right. \\
& \left. + \sum_{l=0}^{k-1} \frac{4096 s_l^5 \rho_l (n_l+1)^2 (n_{l+1}+1)\mu}{|\sigma^{(l)}\sigma^{(l+1)}|}\right) \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
& + O(\mu^2) \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad (5.42)
\end{aligned}
$$

$$
\begin{aligned}
|\mathcal{E}_3^{(k)}| \;\leq\; & \left(\frac{8.08(n_{k+1}+1)(n_k+1)(s_k+1)^2 \mu}{|\sigma^{(k)}|}\right. \\
& \left. + \sum_{l=0}^{k-1} \frac{32.32(n_{l+1}+1)^3 (s_l+1)^2 (n_l+1)\mu}{|\sigma^{(l)}\sigma^{(l+1)}|}\right) \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
& + O(\mu^2) \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad (5.43)
\end{aligned}
$$

Proof These results follow from the bounds obtained in Section 5.2 and Theorem 5.15. We derive the bound for $\mathcal{E}_2^{(k)}$ only (defined explicitly in (5.26)); the other two follow by even simpler arguments. From (5.26), (5.5) , (5.11), (5.39), and the fact that

$$
\begin{pmatrix} max\{\| \hat{P}^{(i)} \|, \| \hat{Q}^{(i)} \|\} \\ max\{\| \hat{U}^{(i)} \|, \| \hat{V}^{(i)} \|\} \end{pmatrix} \leq |\mathcal{G}_i^{(i)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad (5.44)
$$

$$
\begin{aligned}
|\mathcal{E}_2^{(k)}| \;\leq\; & |\mathcal{C}_2^{(k)}| + \sum_{l=0}^{k-1} |\mathcal{G}_{l+1}^{(k)} \mathcal{C}_2^{(l)}| \leq |\mathcal{C}_2^{(k)}| + \sum_{l=0}^{k-1} 2 s_l |\mathcal{G}_{l+1}^{(k)}||\mathcal{C}_2^{(l)}| \\
\leq\; & (128 s_k^4 \rho_k (n_k+1)\mu + O(\mu^2)) \begin{pmatrix} max\{\| \tilde{P}^{(k)} \|, \| \hat{Q}^{(k)} \|\} \\ max\{\| \hat{U}^{(k)} \|, \| \hat{V}^{(k)} \|\} \end{pmatrix}
\end{aligned}
$$

$$+\sum_{l=0}^{k-1}(256s_l^5\rho_l(n_l+1)\mu+O(\mu^2))|\mathcal{G}_{l+1}^{(k)}|\begin{pmatrix} max\{\|\hat{P}^{(l)}\|,\|\hat{Q}^{(l)}\|\} \\ max\{\|\hat{U}^{(l)}\|,\|\hat{V}^{(l)}\|\}\end{pmatrix}$$

$$\leq (128s_k^4\rho_k(n_k+1)\mu+O(\mu^2))|\mathcal{G}_k^{(k)}|\begin{pmatrix}1\\1\end{pmatrix}$$

$$+\sum_{l=0}^{k-1}(256s_l^5\rho_l(n_l+1)\mu+O(\mu^2))|\mathcal{G}_{l+1}^{(k)}||\mathcal{G}_l^{(l)}|\begin{pmatrix}1\\1\end{pmatrix}$$

$$\leq \frac{512s_k^4\rho_k(n_k+1)^2\mu+O(\mu^2)}{|\sigma^{(k)}|}\begin{pmatrix}1\\1\end{pmatrix}$$

$$+\sum_{l=0}^{k-1}(256s_l^5\rho_l(n_l+1)\mu+O(\mu^2))\frac{16(n_l+1)(n_{l+1}+1)+O(\mu^2)}{|\sigma^{(l)}\sigma^{(l+1)}|}\begin{pmatrix}1\\1\end{pmatrix}$$

and the result follows. ∎

In the above lemma, the bounds obtained involve the products $\sigma^{(l)}\sigma^{(l+1)}$. These result from inequalities involving the expression $|\mathcal{G}_{l+1}^{(k)}||\mathcal{G}_l^{(l)}|$. However, $\mathcal{G}_{l+1}^{(k)}\mathcal{G}_l^{(l)}=\mathcal{G}_l^{(k)}$, so it is felt that the inequalities are crude and the bounds should just involve a single $\sigma^{(l)}$. The experimental results presented in Chapter 6 support this suggestion, but there has been no success yet analytically.

Finally, we have:

Theorem 5.17 *Given the assumptions of Theorem 5.15,*

$$|\mathcal{D}^{(k+1)}| \leq \left(\frac{\Gamma_0(k)}{|\sigma^{(k)}|} + \sum_{l=0}^{k-1} \frac{\Gamma_1(l)}{|\sigma^{(l)}\sigma^{(l+1)}|}\right) \mu \begin{pmatrix} 1 \\ 1 \end{pmatrix} + O(\mu^2) \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tag{5.45}$$

where

$$\Gamma_0(l) = 4.04(s_l+1)(n_l+1)^3 + 512 s_l^4 \rho_l (n_l+1)^2 + 8.08(n_{l+1}+1)(n_l+1)(s_l+1)^2 \text{ and} \tag{5.46}$$

$$\Gamma_1(l) = 32.32 s_l (s_l+1)(n_l+1)^3(n_{l+1}+1) + 4096 s_l^5 \rho_l (n_l+1)^2(n_{l+1}+1) + 32.32(n_{l+1}+1)^3(s_l+1)^2(n_l+1). \tag{5.47}$$

Proof Since $\mathcal{D}^{(k+1)} = \mathcal{E}_1^{(k)} + \mathcal{E}_2^{(k)} + \mathcal{E}_3^{(k)}$, application of Lemma (5.16) yields

$$\begin{aligned} |\mathcal{D}^{(k+1)}| \leq\ & \left(4.04(s_k+1)(n_k+1)^3 + 512 s_k^4 \rho_k (n_k+1)^2 \right. \\ & \left. + 8.08(n_{k+1}+1)(n_k+1)(s_k+1)^2\right) \frac{\mu}{|\sigma^{(k)}|} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ & + \sum_{l=0}^{k-1} \left(\left(32.32 s_l (s_l+1)(n_l+1)^3(n_{l+1}+1) + 4096 s_l^5 \rho_l (n_l+1)^2(n_{l+1}+1) \right. \right. \\ & \left. \left. + 32.32(n_{l+1}+1)^3(s_l+1)^2(n_l+1)\right) \frac{\mu}{|\sigma^{(l)}\sigma^{(l+1)}|}\right) \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ & + O(\mu^2) \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{aligned}$$

and the result follows. ∎

Observe that each term in the summation in (5.45) is independent of k. Also, Theorem 5.17 assures us that if $|\mathcal{D}^{(k)}|$ is small and $|r_0^{(k)}|$ and $|v_0^{(k)}|$ are large, then $|\mathcal{D}^{(k+1)}|$ will also be small. Moreover, if s_k is chosen so that $|r_0^{(k+1)}|$ and $|v_0^{(k+1)}|$ are large as well, then Theorem 5.17 can be applied again at the next iteration. Thus, $|\mathcal{D}^{(k)}|$ will remain small for all k, as long as s_k can be chosen in such a fashion. In the next chapter, we give experimental results that show that s_k can be so chosen. Now, however, we extend the above results to give bounds on the errors in the computed Padé approximants.

5.4 Relative Errors and Stability

At the end of the k^{th} step, the algorithm has computed $(P^{(k)}, Q^{(k)})$ and $(U^{(k)}, V^{(k)})$. These are not, however, the true solutions $({P^{(k)}}_{TRUE}, {Q^{(k)}}_{TRUE})$ and $({U^{(k)}}_{TRUE}, {V^{(k)}}_{TRUE})$, where the true solutions are normalized so that $v_0^{(k)} = v_{0,TRUE}^{(k)}$ and $r_0^{(k)} = r_{0,TRUE}^{(k)}$[5]. We define the error in the computed Padé approximants to be

$$(\delta P^{(k)}, \delta Q^{(k)}) = (P^{(k)}, Q^{(k)}) - ({P^{(k)}}_{TRUE}, {Q^{(k)}}_{TRUE}) \tag{5.48}$$

[5]This particular normalization was chosen to simplify the following analysis.

and

$$(\delta U^{(k)}, \delta V^{(k)}) = (U^{(k)}, V^{(k)}) - ({U^{(k)}}_{TRUE}, {V^{(k)}}_{TRUE}). \tag{5.49}$$

We now use the order conditions

$$AQ^{(k)} - P^{(k)} = \delta R^{(k)} + z^{2n} R^{(k)} \tag{5.50}$$

$$AV^{(k)} - U^{(k)} = \delta W^{(k)} + z^{2n} W^{(k)} \tag{5.51}$$

and the results obtained in the previous section for

$$\mathcal{D}^{(k)} = \begin{pmatrix} \delta R^{(k)} \\ \delta W^{(k)} \end{pmatrix} \tag{5.52}$$

to obtain expressions for and bounds on $(\delta P^{(k)}, \delta Q^{(k)})$ and $(\delta U^{(k)}, \delta V^{(k)})$. Once this is done, we are then able to provide a bound for the relative errors

$$\frac{\| \delta P \| + \| \delta Q \|}{\| P \| + \| Q \|} \text{ and } \frac{\| \delta U \| + \| \delta V \|}{\| U \| + \| V \|}.$$

We shall see that the bounds depend critically upon $|\mathcal{D}^{(k)}|$ and $\sigma^{(k)}$. Once this is done, we are in a position to make statements about the stability of the algorithm.

For the moment, we dispense with the superscripts on the quantities being discussed in order to simplify the notation. The superscripts will reappear when they are needed again for clarity. Letting

$$H_n = H_{n-1,n} \text{ and } J_n = \begin{pmatrix} 0 & & a_0 \\ & \iddots & \vdots \\ a_0 & \cdots & a_n \end{pmatrix},$$

the order conditions (5.50) and (5.51) give us

$$H_n \begin{pmatrix} v_n \\ \vdots \\ v_1 \end{pmatrix} = -v_0 \begin{pmatrix} a_n \\ \vdots \\ a_{2n-1} \end{pmatrix} + \begin{pmatrix} \delta w_n \\ \vdots \\ \delta w_{2n-1} \end{pmatrix} \tag{5.53}$$

$$J_{n-1} \begin{pmatrix} v_{n-1} \\ \vdots \\ v_0 \end{pmatrix} = \begin{pmatrix} u_0 \\ \vdots \\ u_{n-1} \end{pmatrix} + \begin{pmatrix} \delta w_0 \\ \vdots \\ \delta w_{n-1} \end{pmatrix} \tag{5.54}$$

$$H_n \begin{pmatrix} q_{n+1} \\ \vdots \\ q_2 \end{pmatrix} = r_o e_n + \begin{pmatrix} \delta r_{n+1} \\ \vdots \\ \delta r_{2n-1} \\ 0 \end{pmatrix} \tag{5.55}$$

$$J_{n-2} \begin{pmatrix} q_n \\ \vdots \\ q_2 \end{pmatrix} = \begin{pmatrix} p_2 \\ \vdots \\ p_n \end{pmatrix} + \begin{pmatrix} \delta r_2 \\ \vdots \\ \delta r_n \end{pmatrix} \tag{5.56}$$

keeping in mind that the constant and linear terms in P, Q, and δR are zero.

From these equations, it is easy to see that the following is true.

Lemma 5.18

$$\begin{pmatrix} \delta v_n \\ \vdots \\ \delta v_0 \end{pmatrix} = \begin{pmatrix} H_n^{-1} \begin{pmatrix} \delta w_n \\ \vdots \\ \delta w_{2n-1} \end{pmatrix} \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \delta u_0 \\ \vdots \\ \delta u_{n-1} \end{pmatrix} = J_{n-1} \begin{pmatrix} \delta v_{n-1} \\ \vdots \\ \delta v_0 \end{pmatrix} - \begin{pmatrix} \delta w_0 \\ \vdots \\ \delta w_{n-1} \end{pmatrix}$$

$$\begin{pmatrix} \delta q_{n+1} \\ \vdots \\ \delta q_2 \end{pmatrix} = H_n^{-1} \begin{pmatrix} \delta r_{n+1} \\ \vdots \\ \delta r_{2n-1} \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \delta p_2 \\ \vdots \\ \delta p_n \end{pmatrix} = J_{n-2} \begin{pmatrix} \delta q_n \\ \vdots \\ \delta q_2 \end{pmatrix} - \begin{pmatrix} \delta r_2 \\ \vdots \\ \delta r_n \end{pmatrix}.$$

Proof These relations follows immediately from the equations (5.53)-(5.56) and the fact that $r_0 = r_{0,TRUE}$ and $v_0 = v_{0,TRUE}$. ∎

With these expressions, we now obtain bounds on $\| \delta P \|$, $\| \delta Q \|$, $\| \delta U \|$, and $\| \delta V \|$. Immediately, from the preceding lemma, we have:

Corollary 5.19

$$\| \delta P \| + \| \delta Q \| \leq (n \| H_n^{-1} \| + 1) \| \delta R \| \text{ and} \tag{5.57}$$

$$\| \delta U \| + \| \delta V \| \leq ((n+1) \| H_n^{-1} \| + 1) \| \delta W \|. \tag{5.58}$$

We have bounds for $\| \delta R \|$ and $\| \delta W \|$, so all that is needed to turn (5.57) and (5.58) into a definitive statement, is a bound for $\| H_n^{-1} \|$. To do

this, we now draw upon the results of Chapter 4 to obtain the following.

Lemma 5.20 *If*

$$22(n+1)^2(\| \delta R \| + \| \delta W \| + 5(n+1)\mu) \leq |r_0 v_0|, \tag{5.59}$$

then

$$\| H_n^{-1} \| \leq \frac{2.2n^2}{|r_0 v_0|}. \tag{5.60}$$

Proof From (4.13), we have that

$$H_n \left(\frac{1}{r_0 v_0} \tilde{H}_n \right) = I + \frac{1}{r_0 v_0} E,$$

where, using (4.11), (5.2), and (5.3),

$$\| \tilde{H}_n \| \leq 2n^2, \tag{5.61}$$

and, from (4.12),

$$\| \tfrac{1}{r_0 v_0} E \| \leq \frac{2n(n+1)(\| \delta R \| + \| \delta W \|)}{|r_0 v_0|}. \tag{5.62}$$

Applying (5.59), which is the same assumption as used in Lemma 5.13, we have that

$$\| \tfrac{1}{r_0 v_0} E \| \leq \frac{1}{11} < 1. \tag{5.63}$$

Since (5.63) holds, Lemma 4.4 and (5.61) give

$$H_n^{-1} = \left(\frac{\tilde{H}_n}{r_0 v_0} \right) (I + \frac{1}{r_0 v_0} E)^{-1}$$

and

$$\begin{aligned} \| H_n^{-1} \| &\leq \left(\frac{\| \tilde{H}_n \|}{|r_0 v_0|} \right) \| (I + \tfrac{1}{r_0 v_0} E)^{-1} \| \\ &\leq \left(\frac{2n^2}{|r_0 v_0|} \right) (1.1) \end{aligned}$$

and the result follows. ∎

With this last lemma, we can now state our principal theorem.

Theorem 5.21 *If*

$$22(n_i+1)^2(\| \delta R^{(i)} \| + \| \delta W^{(i)} \| + 5(n_i+1)\mu) \leq |r_0^{(i)} v_0^{(i)}| \tag{5.64}$$

holds for $0 \leq i \leq k$, *then*

$$\begin{pmatrix} \| \delta P^{(k)} \| + \| \delta Q^{(k)} \| \\ \| \delta U^{(k)} \| + \| \delta V^{(k)} \| \end{pmatrix}$$

$$\leq \frac{2.2(n_k+1)^3}{|\sigma^{(k)}|} \left(\frac{\Gamma_0(k-1)}{|\sigma^{(k-1)}|} + \sum_{l=0}^{k-2} \frac{\Gamma_1(l)}{|\sigma^{(l)}\sigma^{(l+1)}|} \right) \mu \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$+ O(\mu^2) \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tag{5.65}$$

where Γ_0 *and* Γ_1 *are as defined in (5.46) and (5.47).*

Proof This is immediate from Theorem 5.17, Corollary 5.19, and Lemma 5.20, with the relaxation of some of the inequalities even further. ∎

Since $\| P^{(k)} \| + \| Q^{(k)} \| = \| U^{(k)} \| + \| V^{(k)} \| = 1$, Theorem 5.21 gives a bound on the relative error in the computed Padé approximants. In Theorem 5.21, $\sigma^{(k)}$ appears in just the first factor of (5.65); this factor comes from the condition number of the Hankel matrix that corresponds to the Padé approximant that is finally output by NPADE. The other factor in (5.65) comes from the bound on $\mathcal{D}^{(k)}$ and does not involve $\sigma^{(k)}$, the last one that is computed by the algorithm. Thus, if the algorithm computes its last Padé approximant using an unstable block, the residual of the final Padé approximant will be small, even though the relative error in $(U^{(k)}, V^{(k)})$ may not be small.

Corollary 5.22 *If (5.64) holds for $0 \leq i \leq k$, then the approximants computed by NPADE satisfy*

$$\begin{pmatrix} \| \delta P^{(k)} \| + \| \delta Q^{(k)} \| \\ \| \delta U^{(k)} \| + \| \delta V^{(k)} \| \end{pmatrix}$$

$$\leq \frac{2.2(n_k+1)^3}{\varepsilon^3} \left(\Gamma_0(k-1) + \sum_{l=0}^{k-2} \Gamma_1(l) \right) \mu \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$+ O(\mu^2) \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tag{5.66}$$

where Γ_0 and Γ_1 are as defined in (5.46) and (5.47) and ε is the stability parameter used by NPADE.

Proof The algorithm selects s_{i-1} so that $|r_0^i v_0^i| = |\sigma^{(i)}| > \varepsilon$. ∎

The Γ_i have large constants and powers of n_i and s_i in them. We believe that these large values are a consequence of the method of analysis and are not a feature of the actual problem being solved. Thus, the bound is crude, but we believe that it still reflects the behavior of the error. This is discussed further in the next chapter. Also, as mentioned earlier, it is believed that the $\sigma^{(i)}$'s in the bound on $\mathcal{D}^{(i)}$, should be linear and not quadratic in the denominator of the bound. In this case, (5.66) will only involve ε^{-2} and not ε^{-3}. Again, this is discussed at greater length in Chapter 6.

We can now state the principal result of this thesis.

Theorem 5.23 *NPADE is weakly stable over the general class of power series.*

Proof The correspondence between the Padé approximant problem and Hankel matrices tells us that the Padé approximant problem is well-conditioned if and only if the corresponding Hankel matrix is well-conditioned. If the problem is well-conditioned, then $|r_0 v_0|$ must be large for the final Padé approximant that is computed by NPADE; all such intermediate values are large as well due to the choice of s_i by NPADE. But then, by Corollary 5.22, $\| \delta U \| + \| \delta V \|$ will be small. ∎

Chapter 6

Experimental Results

In this chapter, we present some numerical experiments that illustrate the behavior of the algorithm in practice and contrast it with the analysis of the previous section. To put the previous analysis and the following results into context, we present a quote from the 1970 John von Neuman lecture given by James H. Wilkinson [59, page 567]:

> The main object of such an analysis is to expose the potential instabilities, if any, of an algorithm so that hopefully from the insight thus obtained one may be led to improved algorithms. Usually the bound itself is weaker than it might have been because of the necessity of restricting the mass of detail to a reasonable level and because of the limitations imposed by expressing the errors in terms of matrix norms. *A priori* bounds are not, in

general, quantities that should be used in practice. Practical error bounds should usually be determined by some form of *a posteriori* error analysis, since this takes full advantage of the statistical distribution of rounding errors and of any special features, such as sparseness, in the matrix.

The results that we present in this chapter illustrate the behavior of the NPADE algorithm with respect to instabilities in the power series at intermediate points along the superdiagonal of the Padé table. Three random power series were used and they are given in the Appendix. They were chosen as they contain instabilities[1]. The results are presented in Tables 6.1–6.9. The first six tables deal with the behavior of $\mathcal{D}^{(k)}$ and are grouped in pairs, with one pair for each test case. The first in each pair shows the results obtained when NPADE is applied to a test case using $\varepsilon = 10^{-2}$ so that unstable blocks are stepped over. The second table shows the results when $\varepsilon = 10^{-12}$ and unstable blocks are not stepped over. The last three tables (Tables 6.7–6.9) compare the computed results for both choices of ε with the exact results obtained using an algebraic method for each of the three test cases. In all nine tables, $< \cdot >$ is used to denote the sum of the norms of the components of vector and matrix quantities.

[1]The other test cases given in the Appendix are convergent series that mostly have no instabilities.

We shall see that, in practice, the large constants and powers of n_l and s_l that occur in the bounds derived in Chapter 5 are not manifested in the actual results. We shall also see that $\mathcal{D}^{(k)}$ depends on $|\sigma^{(l)}|$ and not $|\sigma^{(l)}\sigma^{(l+1)}|$. With these observations, we present a simpler, and apparently more realistic, bound for $|\mathcal{D}^{(k)}|$. In this form, it is clear that errors from each step are additive and depend upon the smallest $|\sigma^{(l)}|$ that is encountered; this behavior is illustrated in the examples. We also make some observations about how $< \mathcal{G}_k^{(k)} >$ and related quantities affect $< \mathcal{D}^{(k)} >$.

In the discussion of the errors in δU and δV, we use the new bound on $\mathcal{D}^{(k)}$ to describe these errors in the computed Padé approximants. We see that if the smallest $|\sigma^{(l)}|$ appears in the factor that corresponds to the condition number of the Hankel matrix of the problem, then its effect is more pronounced as this is a multiplicative factor. We also observe some behavior in large unstable blocks that is not explained by the bounds of Chapter 5. This behavior does not contradict the results of Chapter 5; rather, in the large blocks the actual error is not as large as expected. Some suggestions about why this is happening are made and this matter is discussed further in Chapter 8. We conclude our examination of the experimental results by observing that $\mathcal{D}^{(k)}$ can be used to give an *a posteriori* estimate of the error in the computed Padé approximants.

The NPADE algorithm was coded in C using double precision. All experiments were run on a Sun SPARCstation 1 which has a unit roundoff of

approximately 10^{-16}.

6.1 The Behavior of δR and δW

We first examine the behavior of $< \mathcal{D}^{(k)} >= \| \delta R^{(k)} \| + \| \delta W^{(k)} \|$. These results are shown in Tables 6.1–6.6. The bounds derived in the previous chapter contained large constants and powers of n_l and s_l. We suggest that these are artifacts resulting from the necessity of simplifying the expressions involving the infinity norm and are not operative factors in the actual errors that occur. Tables 6.1, 6.3, and 6.5, illustrate this. In these examples, s_k is selected so that $|\sigma^{(k)}|$ exceeds 10^{-2}. From $n = 7$ to $n = 30$, there is little or no growth in $< \mathcal{D} >$, but the bounds would suggest a growth of $O(n^4)$, for example. This observation supports our contention that the appearance of n_l in the bounds is due mostly to the method of the analysis and not the problem itself. The stepsize, s_l, arises in the bounds through similar considerations. Thus, we will assume, as is common (Higham and Higham [31]), that the effect of n_l and s_l is linear. With this operational assumption, Theorem 5.17 takes the form

$$|\mathcal{D}^{(k+1)}| \leq \left(n_k M_0 \rho_k \frac{1}{|\sigma^{(k)}|} + \sum_{l=0}^{k-1} \frac{n_l M_1 \rho_l}{|\sigma^{(l)} \sigma^{(l+1)}|} \right) \mu \begin{pmatrix} 1 \\ 1 \end{pmatrix} + O(\mu^2) \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tag{6.1}$$

where M_0 and M_1 are moderate (approximately $O(10)$) constants.

Another comment that was made in the course of the analysis of the previous chapter, was that it was likely that the product $|\sigma^{(l)}\sigma^{(l+1)}|$ was due to the method of analysis and not due to the problem itself. If any of the larger unstable blocks are examined, for example the one that occurs for $n = 15, 16, 17$ in Table 6.4, we see that the error introduced depends on $|\sigma^{(l)}|$ and not $|\sigma^{(l)}\sigma^{(l+1)}|$. Applying this observation to (6.1) gives us:

$$|\mathcal{D}^{(k+1)}| \leq \left(\sum_{l=0}^{k} \frac{n_l M \rho_l}{|\sigma^{(l)}|}\right) \mu \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix} + O(\mu^2) \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix}. \tag{6.2}$$

where M is a moderate (approximately $O(10)$) constant.

From (6.2), a few things become apparent. Firstly, the bound primarily depends upon $|\sigma^{(l)}|$. Also, we see that the errors introduced in each step are reflected in the total error in an additive fashion; this is really the most important observation that results from the error analysis. To provide some understanding of why this is so, recall that in Lemma 5.11, we have

$$\mathcal{D}^{(k)} = \mathcal{L}^{(k-1)} + \hat{\mathcal{P}}^{(k-1)}\mathcal{D}^{(k-1)}.$$

Applying this iteratively, we obtain

$$\mathcal{D}^{(k)} = \mathcal{L}^{(k-1)} + \mathcal{G}_{k-1}^{(k-1)}\mathcal{L}^{(k-2)} + \mathcal{G}_{k-2}^{(k-1)}\mathcal{L}^{(k-3)} + \cdots + \mathcal{G}_{1}^{(k-1)}\mathcal{L}^{(0)}.$$

This shows us that the error of each step is additive, but as propagated by $\mathcal{G}_i^{(k-1)}$. Now, $\mathcal{G}_i^{(k-1)}$ appears to be dependent upon k, but this is not so.

The key step in the previous analysis was showing that $|\mathcal{G}_i^{(k)}|$, to order μ, is independent of k. In particular, we have (from Theorem 5.15) that

$$|\mathcal{G}_i^{(k)}| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \leq \frac{4(n_i+1)+O(\mu)}{|\sigma^{(i)}|} \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

It is this fact that makes the errors effectively additive. Also, it shows that the error is bounded by the reciprocal of the smallest $|\sigma^{(l)}|$ encountered.

If we look at Table 6.1, we see that $|\sigma^{(k)}|$ is bounded throughout by 10^{-2}, and because of this there is no growth in the error after the first few terms; this is also seen in Tables 6.3 and 6.5. In Table 6.2, where ε has been relaxed to 10^{-12}, some small $|\sigma^{(i)}|$ appear. At $n = 9$, $|\sigma^{(9)}| = 10^{-8}$, which is approximately 10^6 times larger than any previous $|\sigma^{(l)}|$ and, at the next step, the error grows by 10^5. At $n = 12$, the error grows by an order of magnitude as the reciprocal of $|\sigma^{(11)}|$ exceeds the previous largest $1/|\sigma l|$ by a similar amount. $|\sigma^{(13)}|$, $|\sigma^{(16)}|$, and $|\sigma^{(21)}|$ are also small, but none is smaller than $|\sigma^{(11)}|$, so there is no growth in the error.

In Table 6.4, a small ε is used again. This series is similar to the previous one, except that a few coefficients have been perturbed to introduce a larger instability at the twentieth node. Here, $|\sigma^{(20)}|$ is 10^{-11} and since it dominates the previous $|\sigma^{(i)}|$'s, an increase in the error occurs at $n = 21$. The third series is similar to the first two, except that it has been perturbed to make the instability even more pronounced. At $n = 19$, $|\sigma^{(19)}|$ is 10^{-12} and remains small until $n = 23$. Inside this unstable block, the error does not grow, but

as soon as the block is left, the error grows by a factor of 10^4. This is the difference in magnitude between $|\sigma^{(9)}|$ and $|\sigma^{(22)}|$.

One last comment about the effect of $\sigma^{(l)}$ on $< \mathcal{D}^{(k)} >$ is in order. If we look at large unstable blocks, such as the one just considered in Table 6.6, we see that inside the block $\mathcal{D}^{(k)}$ does not grow. This behavior is not predicted by the analysis; however, it does not contradict the analysis. There is further discussion on this matter in Chapter 8.

A couple observations can be made about the last three columns of the tables. When there is an increase in the error, both the local error and the propagated error ($< \mathcal{L}^{(k-1)} >$ and $< \mathcal{D}_{prop}^{(i-1)} >=< \hat{\mathcal{P}}^{(k-1)}\mathcal{D}^{(k-1)} >$, respectively) contribute to the growth. Otherwise, the local error is approximately unit roundoff. Also, when the error grows, $< \mathcal{G}_{k-1}^{(k-1)} >$ is large. Since

$$\mathcal{P}^{(k)} = \mathcal{G}_i^{(k-1)}\mathcal{P}^{(i)} + O(\mu)$$

and $\mathcal{P}^{(k)}$ and $\mathcal{P}^{(k+1)}$ are of norm one, we see that significant cancellation must be occurring and this loss of significance (due to the cancellation of terms) is the source of some of the error.

6.2 Errors in the Padé Approximants

While the errors in the order condition are important, what is of paramount interest is how the computed $U(z)$ and $V(z)$ compare with the the true $U(z)$

n_k	s_{k-1}	$< \mathcal{D}^{(k)} >$	$\lvert\sigma^k\rvert$	$< \mathcal{L}^{(k-1)} >$	$< \mathcal{D}^{(k-1)}_{prop} >$	$< \mathcal{G}^{(k-1)}_{k-1} >$
2	2	5.898e-17	4.223e-01	5.898e-17	0.000e+00	2.000e+00
3	1	1.110e-16	1.440e-01	4.968e-17	1.112e-16	2.345e+00
4	1	1.665e-16	1.240e-01	1.131e-16	1.131e-16	2.857e+00
5	1	1.943e-16	3.099e-02	7.028e-17	2.368e-16	2.237e+00
6	1	6.939e-16	8.118e-02	6.106e-16	2.944e-16	7.086e+00
7	1	1.776e-15	1.894e-01	2.489e-16	1.703e-15	5.285e+00
8	1	2.998e-15	2.628e-01	1.874e-16	2.964e-15	3.836e+00
10	2	4.111e-15	3.172e-01	2.670e-16	4.188e-15	3.615e+00
12	2	4.954e-15	2.927e-01	4.025e-16	4.906e-15	2.489e+00
14	2	5.931e-15	2.447e-01	3.557e-16	6.046e-15	3.341e+00
17	3	4.732e-15	2.975e-02	4.624e-16	4.749e-15	1.840e+00
18	1	5.039e-15	2.145e-01	1.071e-15	4.414e-15	1.684e+01
19	1	6.773e-15	1.293e-01	1.844e-16	6.807e-15	2.668e+00
23	4	7.979e-15	1.030e-01	5.349e-16	8.041e-15	2.542e+00
24	1	7.310e-15	1.492e-01	2.461e-16	7.300e-15	4.260e+00
25	1	7.690e-15	7.601e-02	3.466e-16	7.748e-15	2.589e+00
26	1	9.094e-15	2.801e-02	1.463e-16	9.091e-15	1.843e+00
27	1	8.558e-15	1.222e-01	6.554e-16	8.172e-15	1.067e+01
28	1	7.265e-15	1.724e-01	2.776e-16	7.372e-15	4.679e+00
29	1	7.105e-15	4.828e-02	2.745e-16	6.979e-15	2.326e+00
30	1	7.272e-15	5.095e-02	3.237e-16	7.237e-15	2.582e+00

Table 6.1: Test Case 1: $\varepsilon = 10^{-2}$

n_k	s_{k-1}	$< \mathcal{D}^{(k)} >$	$\lvert\sigma^k\rvert$	$< \mathcal{L}^{(k-1)} >$	$< \mathcal{D}_{prop}^{(k-1)} >$	$< \mathcal{G}_{k-1}^{(k-1)} >$
2	2	5.898e-17	4.223e-01	5.898e-17	0.000e+00	2.000e+00
3	1	1.110e-16	1.440e-01	4.968e-17	1.112e-16	2.345e+00
4	1	1.665e-16	1.240e-01	1.131e-16	1.131e-16	2.857e+00
5	1	1.943e-16	3.099e-02	7.028e-17	2.368e-16	2.237e+00
6	1	6.939e-16	8.118e-02	6.106e-16	2.944e-16	7.086e+00
7	1	1.776e-15	1.894e-01	2.489e-16	1.703e-15	5.285e+00
8	1	2.998e-15	2.628e-01	1.874e-16	2.964e-15	3.836e+00
9	1	3.646e-15	5.165e-08	1.111e-16	3.664e-15	2.000e+00
10	1	7.231e-10	3.172e-01	6.378e-10	6.827e-10	1.228e+07
11	1	1.446e-09	1.951e-09	1.120e-16	1.446e-09	2.000e+00
12	1	2.628e-08	2.927e-01	2.556e-08	1.758e-08	3.000e+08
13	1	5.114e-08	3.095e-06	2.024e-16	5.114e-08	2.001e+00
14	1	6.787e-08	2.447e-01	1.605e-11	6.787e-08	1.581e+05
15	1	8.571e-08	6.627e-04	2.467e-16	8.571e-08	1.938e+00
16	1	8.682e-08	2.396e-07	1.266e-16	8.682e-08	2.001e+00
17	1	9.160e-08	2.975e-02	1.681e-11	9.161e-08	2.481e+05
18	1	8.370e-08	2.145e-01	1.763e-15	8.370e-08	1.684e+01
19	1	6.949e-08	1.293e-01	2.841e-16	6.949e-08	2.668e+00
20	1	7.003e-08	6.928e-03	2.668e-16	7.003e-08	1.897e+00
21	1	7.200e-08	6.825e-06	1.515e-16	7.200e-08	1.994e+00
22	1	7.274e-08	4.860e-03	9.703e-14	7.274e-08	1.429e+03
23	1	8.379e-08	1.030e-01	3.943e-15	8.379e-08	4.308e+01
24	1	1.291e-07	1.492e-01	3.574e-16	1.291e-07	4.260e+00
25	1	1.656e-07	7.601e-02	2.191e-16	1.656e-07	2.589e+00
26	1	1.493e-07	2.801e-02	1.596e-16	1.493e-07	1.843e+00
27	1	1.305e-07	1.222e-01	1.188e-15	1.305e-07	1.067e+01
28	1	1.241e-07	1.724e-01	2.942e-16	1.241e-07	4.679e+00
29	1	1.166e-07	4.828e-02	3.793e-16	1.166e-07	2.326e+00
30	1	1.321e-07	5.095e-02	1.923e-16	1.321e-07	2.582e+00

Table 6.2: Test Case 1: $\varepsilon = 10^{-12}$

n_k	s_{k-1}	$< \mathcal{D}^{(k)} >$	$\lvert\sigma^k\rvert$	$< \mathcal{L}^{(k-1)} >$	$< \mathcal{D}^{(k-1)}_{prop} >$	$< \mathcal{G}^{(k-1)}_{k-1} >$
2	2	5.898e-17	4.223e-01	5.898e-17	0.000e+00	2.000e+00
3	1	1.110e-16	1.440e-01	4.968e-17	1.112e-16	2.345e+00
4	1	1.665e-16	1.240e-01	1.131e-16	1.131e-16	2.857e+00
5	1	1.943e-16	3.099e-02	7.028e-17	2.368e-16	2.237e+00
6	1	6.939e-16	8.118e-02	6.106e-16	2.944e-16	7.086e+00
7	1	1.776e-15	1.894e-01	2.489e-16	1.703e-15	5.285e+00
8	1	2.998e-15	2.628e-01	1.874e-16	2.964e-15	3.836e+00
10	2	4.111e-15	3.172e-01	2.670e-16	4.188e-15	3.615e+00
12	2	4.954e-15	2.927e-01	4.025e-16	4.906e-15	2.489e+00
14	2	5.931e-15	2.447e-01	3.557e-16	6.046e-15	3.341e+00
17	3	4.732e-15	2.975e-02	4.624e-16	4.749e-15	1.840e+00
18	1	5.039e-15	2.145e-01	1.071e-15	4.414e-15	1.684e+01
19	1	6.773e-15	1.293e-01	1.844e-16	6.807e-15	2.668e+00
21	2	7.528e-15	2.003e-02	3.643e-16	7.592e-15	1.871e+00
23	2	8.550e-15	6.694e-02	1.344e-15	8.047e-15	9.643e+00
24	1	8.810e-15	1.449e-01	3.978e-16	8.704e-15	5.272e+00
25	1	9.702e-15	7.957e-02	2.575e-16	9.661e-15	2.515e+00
27	2	1.057e-14	1.186e-01	9.966e-16	1.078e-14	4.755e+00
28	1	9.279e-15	1.798e-01	4.094e-16	9.242e-15	4.923e+00
29	1	9.272e-15	8.900e-02	3.279e-16	9.266e-15	2.490e+00
30	1	9.680e-15	9.721e-02	3.123e-16	9.682e-15	2.666e+00

Table 6.3: Test Case 2: $\varepsilon = 10^{-2}$

n_k	s_{k-1}	$< \mathcal{D}^{(k)} >$	$\lvert\sigma^k\rvert$	$< \mathcal{L}^{(k-1)} >$	$< \mathcal{D}_{prop}^{(k-1)} >$	$< \mathcal{G}_{k-1}^{(k-1)} >$
2	2	5.898e-17	4.223e-01	5.898e-17	0.000e+00	2.000e+00
3	1	1.110e-16	1.440e-01	4.968e-17	1.112e-16	2.345e+00
4	1	1.665e-16	1.240e-01	1.131e-16	1.131e-16	2.857e+00
5	1	1.943e-16	3.099e-02	7.028e-17	2.368e-16	2.237e+00
6	1	6.939e-16	8.118e-02	6.106e-16	2.944e-16	7.086e+00
7	1	1.776e-15	1.894e-01	2.489e-16	1.703e-15	5.285e+00
8	1	2.998e-15	2.628e-01	1.874e-16	2.964e-15	3.836e+00
9	1	3.646e-15	5.165e-08	1.111e-16	3.664e-15	2.000e+00
10	1	7.231e-10	3.172e-01	6.378e-10	6.827e-10	1.228e+07
11	1	1.446e-09	1.951e-09	1.120e-16	1.446e-09	2.000e+00
12	1	2.628e-08	2.927e-01	2.556e-08	1.758e-08	3.000e+08
13	1	5.114e-08	3.095e-06	2.024e-16	5.114e-08	2.001e+00
14	1	6.787e-08	2.447e-01	1.605e-11	6.787e-08	1.581e+05
15	1	8.571e-08	6.627e-04	2.467e-16	8.571e-08	1.938e+00
16	1	8.682e-08	2.396e-07	1.266e-16	8.682e-08	2.001e+00
17	1	9.160e-08	2.975e-02	1.681e-11	9.161e-08	2.481e+05
18	1	8.370e-08	2.145e-01	1.763e-15	8.370e-08	1.684e+01
19	1	6.949e-08	1.293e-01	2.841e-16	6.949e-08	2.668e+00
20	1	6.797e-08	1.772e-11	1.910e-16	6.797e-08	2.000e+00
21	1	2.132e-07	2.003e-02	2.725e-07	2.430e-07	2.261e+09
22	1	3.489e-07	9.462e-04	1.589e-16	3.489e-07	2.011e+00
23	1	1.412e-06	6.694e-02	9.812e-15	1.412e-06	1.395e+02
24	1	3.477e-06	1.449e-01	4.668e-16	3.477e-06	5.272e+00
25	1	3.811e-06	7.957e-02	2.739e-16	3.811e-06	2.515e+00
26	1	2.837e-06	6.421e-03	2.088e-16	2.837e-06	1.845e+00
27	1	4.120e-06	1.186e-01	4.423e-15	4.120e-06	4.077e+01
28	1	8.604e-06	1.798e-01	4.718e-16	8.604e-06	4.923e+00
29	1	1.140e-05	8.901e-02	3.210e-16	1.140e-05	2.490e+00
30	1	1.033e-05	9.721e-02	4.143e-16	1.033e-05	2.666e+00

Table 6.4: Test Case 2: $\varepsilon = 10^{-12}$

n_k	s_{k-1}	$< \mathcal{D}^{(k)} >$	$\lvert\sigma^k\rvert$	$< \mathcal{L}^{(k-1)} >$	$< \mathcal{D}_{prop}^{(k-1)} >$	$< \mathcal{G}_{k-1}^{(k-1)} >$
2	2	2.862e-17	2.044e-01	2.862e-17	0.000e+00	2.000e+00
3	1	5.551e-17	9.583e-02	1.194e-17	5.556e-17	2.471e+00
4	1	8.327e-17	8.253e-02	2.781e-17	5.546e-17	2.859e+00
5	1	9.714e-17	2.067e-02	5.923e-17	1.147e-16	2.238e+00
6	1	1.943e-16	5.536e-02	7.980e-17	2.130e-16	7.220e+00
7	1	5.135e-16	1.416e-01	1.297e-16	4.715e-16	5.652e+00
8	1	9.714e-16	2.369e-01	1.197e-16	9.599e-16	4.300e+00
10	2	1.499e-15	2.770e-01	2.128e-16	1.512e-15	3.384e+00
11	1	1.499e-15	7.183e-02	9.325e-17	1.497e-15	2.191e+00
12	1	1.700e-15	2.589e-01	5.518e-16	1.668e-15	8.366e+00
13	1	1.908e-15	5.345e-02	1.249e-16	1.919e-15	2.321e+00
14	1	2.471e-15	3.751e-01	7.216e-16	2.705e-15	1.380e+01
15	1	3.096e-15	2.759e-01	1.301e-16	3.101e-15	2.628e+00
16	1	2.876e-15	1.420e-01	8.595e-17	2.904e-15	2.288e+00
17	1	3.254e-15	2.812e-01	1.139e-16	3.216e-15	3.969e+00
18	1	3.889e-15	3.980e-01	1.572e-16	3.911e-15	3.714e+00
23	5	3.775e-15	2.843e-01	3.886e-16	3.770e-15	2.409e+00
24	1	3.109e-15	4.140e-02	1.508e-16	3.259e-15	1.886e+00
25	1	2.831e-15	1.970e-02	2.363e-16	2.794e-15	3.029e+00
26	1	2.942e-15	1.792e-02	2.444e-16	2.698e-15	2.033e+00
27	1	4.136e-15	1.081e-01	1.460e-15	4.288e-15	1.357e+01
28	1	4.524e-15	6.859e-02	2.727e-16	4.584e-15	2.648e+00
29	1	3.671e-15	4.949e-02	2.031e-16	3.650e-15	2.003e+00
30	1	3.324e-15	6.600e-02	4.800e-16	3.143e-15	5.241e+00

Table 6.5: Test Case 3: $\varepsilon = 10^{-2}$

n_k	s_{k-1}	$< \mathcal{D}^{(k)} >$	$\lvert\sigma^k\rvert$	$< \mathcal{L}^{(k-1)} >$	$< \mathcal{D}_{prop}^{(k-1)} >$	$< \mathcal{G}_{k-1}^{(k-1)} >$
2	2	2.862e-17	2.044e-01	2.862e-17	0.000e+00	2.000e+00
3	1	5.551e-17	9.583e-02	1.194e-17	5.556e-17	2.471e+00
4	1	8.327e-17	8.253e-02	2.781e-17	5.546e-17	2.859e+00
5	1	9.714e-17	2.067e-02	5.923e-17	1.147e-16	2.238e+00
6	1	1.943e-16	5.536e-02	7.980e-17	2.130e-16	7.220e+00
7	1	5.135e-16	1.416e-01	1.297e-16	4.715e-16	5.652e+00
8	1	9.714e-16	2.369e-01	1.197e-16	9.599e-16	4.300e+00
9	1	1.249e-15	5.229e-08	9.443e-17	1.221e-15	2.000e+00
10	1	2.326e-10	2.770e-01	4.363e-10	5.004e-10	1.060e+07
11	1	4.987e-10	7.183e-02	1.036e-16	4.987e-10	2.191e+00
12	1	3.859e-10	2.589e-01	5.759e-16	3.859e-10	8.366e+00
13	1	2.938e-10	5.345e-02	1.154e-16	2.938e-10	2.321e+00
14	1	4.977e-10	3.751e-01	5.022e-16	4.977e-10	1.380e+01
15	1	6.286e-10	2.759e-01	9.714e-17	6.286e-10	2.628e+00
16	1	7.481e-10	1.420e-01	1.139e-16	7.481e-10	2.288e+00
17	1	8.092e-10	2.812e-01	1.957e-16	8.092e-10	3.969e+00
18	1	8.188e-10	3.980e-01	2.303e-16	8.188e-10	3.714e+00
19	1	8.928e-10	3.878e-12	1.330e-16	8.928e-10	1.948e+00
20	1	8.412e-10	2.293e-11	1.113e-15	8.412e-10	1.209e+01
21	1	8.412e-10	2.293e-11	1.388e-16	8.412e-10	2.000e+00
22	1	8.928e-10	3.878e-12	1.489e-16	8.928e-10	2.045e+00
23	1	7.185e-06	2.843e-01	1.215e-05	9.325e-06	1.428e+11
24	1	1.629e-05	4.140e-02	1.929e-16	1.629e-05	1.886e+00
25	1	1.719e-05	1.970e-02	3.862e-16	1.719e-05	3.029e+00
26	1	1.746e-05	1.792e-02	8.537e-17	1.746e-05	2.033e+00
27	1	2.651e-05	1.081e-01	1.162e-15	2.651e-05	1.357e+01
28	1	3.322e-05	6.860e-02	3.248e-16	3.322e-05	2.648e+00
29	1	2.593e-05	4.949e-02	1.569e-16	2.593e-05	2.003e+00
30	1	2.408e-05	6.600e-02	6.735e-16	2.408e-05	5.241e+00

Table 6.6: Test Case 3: $\varepsilon = 10^{-12}$

and $V(z)$. To examine this question, the approximants for the test series were computed exactly. By multiplying the coefficients of $A(z)$ in these series by 10^{21}, they were converted to integer power series. The $(29, 30)$ Padé approximants for these series were then computed using the Maple symbolic system with a program called JPADE[2] (Kossowski [37]). The answers so obtained were then converted to the real domain by shifting the decimal point and chopping the coefficients to twenty significant digits. The approximants were then converted to normalized Padé approximants. The differences between the exact and computed polynomials were determined and their norms computed. Tables 6.7–6.9 give the values of $\| \delta U^{(k)} \| + \| \delta V^{(k)} \|$. Two values are given for each n: the stable column gives the error that occurs when $\varepsilon = 10^{-2}$ and the unstable column gives it for $\varepsilon = 10^{-12}$. As predicted by the analysis, the former is much better than the latter.

If we use (6.2), then Theorem 5.21 yields

$$\| \delta U^{(k)} \| + \| \delta V^{(k)} \| \leq \frac{1}{|\sigma^{(k)}|} \left(\sum_{l=0}^{k} \frac{n_l M \rho_l}{|\sigma^{(l)}|} \right) \mu \begin{pmatrix} 1 \\ 1 \end{pmatrix} + O(\mu^2) \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \quad (6.3)$$

This suggests that in any unstable block, the error in the computed $(j-1, j)$ Padé approximant should be large due to the multiplicative effect of $|\sigma^{(j)}|$ in (6.3) and then decrease once the block is left; at this point, the small $|\sigma^{(j)}|$ now appears as one of the terms in the second factor of (6.3) where its effect

[2]This program calculates Padé approximants over an integral domain and was programmed by H. Du.

is now additive. This is best illustrated in Table 6.9, where the error grows very large inside an unstable block and then decreases once the block is left and the corresponding Hankel matrix becomes well-conditioned again (i.e., $|\sigma^{(k)}|$ becomes large once again). Once the block is left, the error is larger than it was prior to entering the block, but not as large as it was inside the block as the effect of the instability is now appearing through $< \mathcal{D}^{(k)} >$.

However, one interesting feature is observed in this example, namely that the error of the last entry in the unstable block is not as large as the other entries in the block. This is also seen, for example, in Table 6.7, where the error in the approximant does not become large inside blocks of size 2 (i.e., when a single isolated unstable point is encountered by NPADE). This is not suggested by the bound given in (6.3), but it is not ruled out either. One observation that can be made is that if the block were truly singular, rather than nearly so (i.e., unstable), then the Padé approximant for the last entry in a block would be unique, while all other entries in the block would be nonunique. It is believed that this behavior is somehow related to the question of approximant uniqueness, but currently no rigorous results have been obtained that actually show this.

We can express Theorem 5.21 in a different form than that given in (6.3), namely

$$\| \delta U \| + \| \delta V \| \leq \frac{K}{\varepsilon}(\| \delta R \| + \| \delta W \|), \tag{6.4}$$

where K is a modest (approximately $O(10)$) constant. If we apply the bound

for $\mathcal{D}^{(k)}$ from (6.2), then, as an *a priori* bound, we have

$$\| \delta U \| + \| \delta V \| \leq \max_{0 \leq i \leq k-1} \frac{Ln_i}{|\sigma^{(k)}\sigma^{(i)}|} \leq \frac{Ln_k}{\varepsilon^2}, \tag{6.5}$$

where L is a modest (approximately $O(10)$) constant and the $(n_k - 1, n_k)$ Padé approximant is computed by NPADE. However, δR and δW are easily computed quantities, so we can actually obtain a better *a posteriori* bound using (6.4). For example, if we examine the results given in this chapter for the stable application of NPADE (i.e., for $\varepsilon = 10^{-2}$), we see that, more or less,

$$< \mathcal{D}^{(k)} > \leq \| \delta U^{(k)} \| + \| \delta V^{(k)} \| \leq 100 < \mathcal{D}^{(k)} > .$$

n_k	Stable	Unstable
2	5.5511e-17	5.5511e-17
3	8.3267e-17	8.3267e-17
4	1.9082e-16	1.9082e-16
5	2.2204e-16	2.2204e-16
6	4.7184e-16	4.7184e-16
7	1.1796e-15	1.1796e-15
8	2.3176e-15	2.3176e-15
9	Unstable	4.2466e-15
10	3.8580e-15	1.1035e-09
11	Unstable	9.1274e-10
12	8.0908e-15	3.1795e-08
13	Unstable	4.1980e-08
14	6.8556e-15	6.0450e-08
15	Unstable	4.2025e-07
16	Unstable	1.0281e-07
17	4.1078e-15	6.1857e-08
18	5.8703e-15	5.2384e-08
19	1.1754e-14	4.5629e-08
20	Unstable	1.9954e-07
21	Unstable	2.4984e-07
22	Unstable	1.4915e-07
23	6.8279e-15	7.0215e-08
24	5.3291e-15	3.1958e-07
25	7.7716e-15	1.5034e-07
26	6.0785e-15	1.3825e-07
27	4.3923e-15	7.5392e-08
28	5.5789e-15	6.5606e-08
29	2.9657e-14	1.1872e-07
30	6.4115e-15	1.8561e-07

Table 6.7: Test Case 1: $\| \delta U^{(k)} \| + \| \delta V^{(k)} \|$

n_k	Stable	Unstable
2	5.5511e-17	5.5511e-17
3	8.3267e-17	8.3267e-17
4	1.9082e-16	1.9082e-16
5	2.2204e-16	2.2204e-16
10	3.8580e-15	1.1035e-09
12	8.0908e-15	3.1795e-08
23	7.2997e-15	1.9575e-06
29	1.5071e-14	1.3193e-05
30	1.4794e-14	1.1820e-05

Table 6.8: Test Case 2: $\| \delta U^{(k)} \| + \| \delta V^{(k)} \|$

n_k	Stable	Unstable
17	4.8087e-15	1.0801e-09
18	4.8572e-15	1.3241e-09
19	Unstable	1.8158e+00
20	Unstable	7.8434e-01
21	Unstable	1.8158e+00
22	Unstable	1.9947e-09
23	5.9397e-15	2.3055e-05
24	4.4791e-15	1.7767e-05
25	5.7176e-15	2.8251e-04
26	6.7793e-15	4.7562e-05
27	2.4702e-14	3.1553e-05
28	5.3513e-14	1.3508e-04
29	1.7014e-14	3.4901e-05
30	4.2882e-15	5.1040e-05

Table 6.9: Test Case 3: $\| \delta U^{(k)} \| + \| \delta V^{(k)} \|$

Chapter 7

Sequence Acceleration

7.1 An Introduction to Sequence Acceleration

In this chapter, we consider a special application of the Padé approximant, namely sequence acceleration. We switch for the moment from series to sequences. For any power series there are two sequences associated with it. One is the sequence of the coefficients of the powers of the variable in the power series. The other is the sequence of partial sums of the power series. These partial sums are evaluated at some value of z so that this sequence consists of real values; the evaluation is usually performed at $z = 1$. We switch back and forth between the two in the following discussion. For the experimental results, we allow the programs to handle the conversion from

one type of sequence to another.

Sequence acceleration refers to the process of taking a sequence $\mathcal{S} = (s_0, s_1, \ldots)$ and applying a transformation T to it, thereby obtaining a new sequence $T\mathcal{S}$. The goal, in general, is to obtain transformed sequences that converge "faster" than the original sequence. The term faster is used advisedly because the original sequence may be divergent. Wimp [62] surveys a large number of linear and nonlinear sequence transformations; Hardy [30] is the classic work dealing with linear transformations. T is usually required to satisfy the following requirements [62]:

(i) T is homogeneous; i.e., $T(\alpha\mathcal{S}) = \alpha T(\mathcal{S})$ for any scalar α;

(ii) T is translative; i.e., $T(\mathcal{S} + \mathcal{C}) = T(\mathcal{S}) + T(\mathcal{C})$ for any constant sequence $\mathcal{C}$;

(iii) T is regular for $\mathcal{S}$; i.e., if $\mathcal{S}$ converges, then $T\mathcal{S}$ converges to the same limit.

Often instead of (iii), the following is required:

(iii') T is accelerative for $\mathcal{S}$; i.e., $T\mathcal{S}$ converges more rapidly than $\mathcal{S}$.

Definition 7.1 *If a sequence $\mathcal{S}$ is divergent, but $T\mathcal{S}$ has a limit, s, we say that s is the* **antilimit** *of $\mathcal{S}$.*

Example 7.2 Consider the sequence $\mathcal{S} = (1, 0, 1, 0, 1, \ldots)$. One sequence transformation of $\mathcal{S}$ is Aitken's δ^2 method which is defined by

$$s_n^T = s_n - \frac{(s_{n+1} - s_n)^2}{(s_{n+2} - 2s_{n+1} + s_n)},$$

where s_n^T denotes the n^{th} term in the transformed sequence $T\mathcal{S}$ (Isaacson and Keller [33]). When T is applied, the result is $T\mathcal{S} = (1/2, 1/2, \ldots)$ and

$1/2$ is taken to be the antilimit of $\mathcal{S}$. This value has significance once one understands where $\mathcal{S}$ comes from. $\mathcal{S}$ is the sequence of partial sums at $z = 1$ for the Maclaurin expansion of $1/(1+z)$; the antilimit $1/2$ is the value of the function at $z = 1$. The Maclaurin expansion does not converge at this point, but the sequence transformation provides a means of analytically continuing the function. ∎

In 1955, Shanks [51] defined a set of nonlinear transforms that he applied to a number of sequences and related these transforms to Padé approximants. This paper "was widely read and made an enormous impact"(Wimp [62, page 120]). Unfortunately, his definition of the transformations required the computation of the determinants of ever larger matrices in order to obtain the transform for a given sequence. Wynn [63] derived an iterative method to compute Shank's transforms. This method is known as the ϵ-algorithm. If $(s_0, s_1, \ldots)$ is the sequence being transformed, then the ϵ-algorithm is defined by

$$\epsilon_{k+1}^{j} = \epsilon_{k-1}^{j+1} + (\epsilon_{k}^{j+1} - \epsilon_{k}^{j})^{-1}, \tag{7.1}$$

where $\epsilon_{-1}^{j} = 0$ and $\epsilon_{0}^{j} = s_j, j = 0, 1, 2, \ldots$. With repeated application of the algorithm, the ϵ-table can be generated, where ϵ_{k}^{j} is the entry in the j^{th} row and k^{th} column.

There is an intimate connection between the ϵ-algorithm and the Padé ta-

ble. If (U, V) is an $(k+j, k)$ Padé form for $A(z)$ and ϵ_{2k}^{j} is obtained by applying the ϵ-algorithm to the sequence $\{\sum_{i=0}^{j} a_i\}_{j=0}^{\infty}$, then $\epsilon_{2k}^{j} = U(1)/V(1)$. This is not surprising as Wynn derived the ϵ-algorithm by proving that a relationship between adjacent entries in the Padé table was valid[1]. An even stronger result was actually shown by Wynn [63]. If we apply the ϵ-algorithm to the sequence $\{\sum_{i=0}^{j} a_i z^i\}_{j=0}^{\infty}$ using polynomial arithmetic, then $\epsilon_{2k}^{j}(z) = U(z)/V(z)$. A major drawback to this method for computing Padé approximants is that the numerators and denominators so generated will have nontrivial common factors.

As stated above, the ϵ-algorithm requires that the series be normal, otherwise the difference in (7.1) will be zero. The identity of Wynn has been extended to deal with nonnormality (Wynn [66], Cordellier [17]). In addition, Wynn [64] shows how to extend the scope of the ϵ-algorithm to include near degenerate cases corresponding to unstable blocks of size two in the Padé table. The identity and method are seldom used in numerical studies comparing various transformations. A limited amount of work has been done on the stability of the ϵ-algorithm (e.g., Wynn [65] and Bultheel and Wuytack [10]). Considerably more work has been done in applying the ϵ-algorithm to certain problems (e.g., Gekeler [25], Kahaner [35], Brezinski and Rieu [8]).

In 1973, Levin [40] defined a set of transforms (e.g., the t, u, and v

[1]This identity is now known as Wynn's identity; it is also known as the missing identity of Frobenius (Gragg [28]).

transforms); he used some of Shank's underlying ideas as his starting point. Shanks observed that many sequences looked like sequences generated by the rule $s_n = s + \sum_{j=1}^{k} a_j t_j^n$; i.e., a constant plus a number of transient terms[2]. What he then did was assume that many sequences could be approximated by expressions of the above form. In particular, the k^{th} order local approximation about s_n was given by $s_r = s_{kn} + \sum_{i=1}^{k} a_{in} q_{in}^r$, for $n-k \leq r \leq n+k, (n \geq k)$. The formula that Shanks gave for his transforms was a determinantal form for s_{kn}, resulting in the sequence $(s_{k,k}, s_{k,k+1}, s_{k,k+2}, \ldots)$. Levin approximated sequences locally in a similar fashion by using $s_r = T_{kn} + R_k(r),\ n \leq r \leq n+k$, where the $R_k(r)$ are functions of r involving k arbitrary constants and the T_{kn} are intended to be an approximation of the limit or antilimit of the sequence $\mathcal{S}$. By using different forms for $R_k(r)$ to approximate the difference between s_r and the limit (or antilimit) of $\mathcal{S}$, Levin derived a number of transforms, including the t, u, and v transforms.

Smith and Ford [52], [53] took the ϵ-algorithm, Levin's u transform, and other sequence transformations and numerically compared their accelerative effects when applied to various classes of (normal) sequences. They concluded ([53]) that in all cases where acceleration was achieved, either the u or ϵ transform was most effective. Other, more limited, studies are given by Luke [43], [44], Graves-Morris [29], and Brezinski [7].

[2]Such a sequence will converge if all $|t_j| < 1$ and will not converge if some $|t_j| \geq 1$. In the case of nonconvergence, the constant s is the antilimit of the sequence.

The numerical Padé algorithm can be used to accelerate the convergence of sequences since it is equivalent to the ϵ-algorithm. In the next section, we investigate the performance of NPADE as a sequence accelerator; because of the work of Smith and Ford, we just compare our algorithm with the ϵ-algorithm. Rather than the standard ϵ-algorithm as stated above, we use an implementation of Wynn's algorithm that handles near degenerate blocks of size two. We would have liked to use a version that handled the general case, but we could not see how to extend it. Perhaps this is not surprising given the comments of Claessens and Wuytack [16] about the difficulty of implementing the nonnormal rules for the Padé table in the most general setting.

7.2 Experimental Results

To investigate the performance of NPADE as a sequence accelerator, it was applied to a number of classical sequences from the literature (Shanks [51], Wynn [64], Gragg [28], Levin [40], Smith and Ford [52], [53]). One problem that arose in this process was the use of ε in the acceptance criteria for solutions. As higher approximates are computed and the sequence converges, the residuals tend to 0, so that $|r_0^{(k)}|$ is an ever decreasing quantity. This means that $|\sigma^{(k)}|$ is decreasing and a fixed ε is no longer suitable for use in the criteria. Examination of the sequences showed that the ratio of $|r_0^{k+1}|$

to $< \hat{S}_k >$ (as defined in (5.7)) was roughly constant. Because of this, the acceptance criteria was altered so that if

$$\frac{|r_0^{(k+1)} v_0^{(k+1)}|}{< \hat{S}_k >} \geq \varepsilon$$

the approximant was accepted. This modification is based on intuition about the problem and has no theoretical justification. However, it performed well and it is felt that the analysis of the earlier chapters can be modified to cover sequence acceleration using a criteria such as this. So far, though, success in this endeavor has not yet been achieved.

Smith and Ford [52], [53] perform extensive studies on the performance of sequence accelerators. In [53], they conclude that the ϵ-algorithm is one of the better performers[3]. In Tables 7.10–7.24, given at the end of this chapter, we present comparative results between NPADE and the ϵ-algorithm. The ϵ-algorithm was coded in double precision in C and generated ϵ_{2n}^0, $n = 2, 3, \ldots, N$. The NPADE code was modified so that it produced the corresponding values; in other words, it computed the (n, n) diagonal Padé approximants and evaluated them at $z = 1.0$. The NPADE parameter ε was usually 10^{-4}, though in one case it was chosen to be 10^{-8}. The experiments were run on a SPARCstation 1.

In the examples presented, the performance of the algorithm was compa-

[3]More precisely, in all cases examined, either the ϵ-algorithm or Levin's u transform was the best accelerator.

rable to that of the ϵ-algorithm. Moreover, all but one of the experiments with the ϵ-algorithm agree with the results reported by Smith and Ford. The exception is the Fourier sine series in Table 7.22, where the convergence is slower. It is believed that this is due to the fact that the results in [53] were based on the average acceleration achieved for a number of Fourier sine series, whereas we have just reported the results for one such series.

These experiments illustrate the usefulness of the algorithm as a sequence accelerator. However, there are significant advantages in using NPADE as opposed to the classical ϵ-algorithm. All the examples we have considered so far involve sequences with no unstable subproblems. To illustrate the advantage of the NPADE algorithm in the more general setting, a sequence (Test Case 4, log(2)) was perturbed to introduce instabilities. The details of the perturbation are given in the Appendix in Table A.29. The results are shown in Table 7.25. In this example, NPADE stepped over the unstable block whereas the ϵ-algorithm failed. Thus we see that NPADE will produce answers in the presence of instabilities, such as in the example just considered, whereas the ϵ-algorithm can fail. This ability to handle instabilities is a major advantage of NPADE.

We conclude this chapter with a brief examination of the complexity of the two algorithms. We assume that the algorithms are used to compute the (N, N) Padé approximant evaluated at $z = 1$. A complexity analysis of the ϵ-algorithm is relatively straight forward. The cost of calculating a

single entry in the ϵ-table is two additions and one division, assuming that all other values are available. To get the entry corresponding to the (N, N) Padé approximant, $2N$ antidiagonals in the ϵ-table must be filled in and the i^{th} antidiagonal contains i entries. Thus, the total number of entries to be computed is given by

$$\sum_{i=1}^{2N} i = N(2N+1),$$

and we see that the ϵ-algorithm is an $O(N^2)$ algorithm with an operation count of $3N(2N+1)$.

On the other hand, the complexity analysis for NPADE, while straightforward, has no simple closed form. The analysis depends upon the size of the steps that are taken in the algorithm and there are places, such as in the solution of $\hat{S}_k$, where certain things can be done to reduce the number of operations performed. Rather than going into a detailed analysis with a precise operation count for NPADE, we provide a quick analysis of the number of operations performed and discuss the behavior in the extreme cases of a totally stable system and a totally unstable system. This is sufficient to illustrate the points we wish to make.

The cost of the algorithm is simply

$$\sum_{i=0}^{k-1} (\text{cost of the } i^{th} \text{ iteration}),$$

where k is such that $n_k = N$. We analyse the cost of a single iteration by examining each of the three steps. Now each iteration is repeated for

$s_i = 1, 2, \ldots$, until a stable subsystem is found. The first step computes the first $2s_i$ terms in $R^{(i)}$ and $W^{(i)}$. Since these terms can be accumulated as s_i is incremented, this is a one time cost of $O(s_i n_i)$ and involves approximately $8s_i n_i$ operations. In the second step, Gaussian elimination is performed on a system of order $2s_i$. In this case, the work done for a smaller s_i cannot be reused. Thus, the cost of this step is approximately

$$\sum_{j=1}^{s_i} \frac{2}{3}(2j)^3 = \frac{4s_i^2(s_i+1)^2}{3}.$$

This is a very rough count of the operations in this step as, given the form of the Sylvester system, the operations performed in this step can be reduced. Once the Sylvester system is solved, we have $\hat{\mathcal{P}}^{(i)}$. A series of polynomial operations are then performed in the last step. The approximate cost of this step is, since it is repeated for each s_i,

$$\sum_{j=1}^{s_i} 8(j+1)(n_i+j+1) = 4(s_i+1)(s_i+2)(n_i + \frac{2s_i+3}{3}) + O(1).$$

So the cost of one iteration is approximately

$$8s_i n_i + \frac{4s_i^2(s_i+1)^2}{3} + 4(s_i+1)(s_i+2)(n_i + \frac{2s_i+3}{3})$$

and, therefore, the algorithm's total cost is roughly

$$\sum_{i=0}^{k-1} 8s_i n_i + \frac{4s_i^2(s_i+1)^2}{3} + 4(s_i+1)(s_i+2)(n_i + \frac{2s_i+3}{3}).$$

If the system is completely unstable (i.e., no subproblem is well-conditioned), then the algorithm keeps trying successively larger s_0's until $s_0 = N$, so that,

since $k = 1$ and $n_0 = 0$, we have a cost of $O(N^4)$ (recall that for the first iteration, no first and third steps need to be performed). This is expected as in this case the algorithm reduces to a sequence of Gaussian eliminations that are performed on larger and larger systems. On the other hand, if the system is completely stable, all the steps are of size one so that $k = N$ and $s_i = 1$ for all i. The previous analysis would then suggest a cost of approximately $4N^2$, but certain savings can be realized in the implementation so that the cost is actually about $2N^2$ (Cabay and Kao [13])[4].

Thus, at worst, the algorithm is $O(N^4)$ and at best $O(N^2)$. Moreover, if the s_i are bounded, the algorithm remains $O(N^2)$. In practice, the algorithm increases its complexity just enough to deal with instabilities as they are encountered. Thus, for a little extra computational cost, NPADE will produce results in cases where the ϵ-algorithm fails.

[4]Cabay and Kao [13] analyse an algebraic algorithm that computes diagonal Padé approximants and compare it to Rissanen's algorithm. Rissanen's algorithm is shown to have an operation count of $3N^2$.

log(2), value = 0.693147180559945		
n	NPADE	ϵ-algorithm
2	6.9230769230769218e-01	6.9230769230769229e-01
3	6.9312169312169303e-01	6.9312169312169314e-01
4	6.9314641744548289e-01	6.9314641744548289e-01
5	6.9314715785304015e-01	6.9314715785304026e-01
6	6.9314717988652796e-01	6.9314717988652796e-01
7	6.9314718054001334e-01	6.9314718054001356e-01
8	6.9314718055935609e-01	6.9314718055935620e-01
9	6.9314718055992786e-01	6.9314718055992797e-01
10	6.9314718055994484e-01	6.9314718055994484e-01
11	6.9314718055994529e-01	6.9314718055994540e-01

Table 7.10: Test Case 4

Leibnitz Series, value = 3.1415926535898		
n	NPADE	ϵ-algorithm
2	3.1372549019607847e+00	3.1372549019607847e+00
3	3.1414634146341465e+00	3.1414634146341469e+00
4	3.1415888250921253e+00	3.1415888250921249e+00
5	3.1415925404465406e+00	3.1415925404465410e+00
6	3.1415926502502463e+00	3.1415926502502458e+00
7	3.1415926534912955e+00	3.1415926534912963e+00
8	3.1415926535868901e+00	3.1415926535868910e+00
9	3.1415926535897092e+00	3.1415926535897092e+00
10	3.1415926535897913e+00	3.1415926535897922e+00
11	3.1415926535897940e+00	3.1415926535897944e+00

Table 7.11: Test Case 5

Alternating Series, value = 0.60489864342162		
n	NPADE	ϵ-algorithm
2	6.0429879839139833e-01	6.0429879839139844e-01
3	6.0488051170197499e-01	6.0488051170197499e-01
4	6.0489810219591922e-01	6.0489810219591933e-01
5	6.0489862735236810e-01	6.0489862735236810e-01
6	6.0489864294584195e-01	6.0489864294584184e-01
7	6.0489864340756572e-01	6.0489864340756572e-01
8	6.0489864342121502e-01	6.0489864342121524e-01
9	6.0489864342161814e-01	6.0489864342161814e-01

Table 7.12: Test Case 6

Divergent Series, value = log(3) = 1.0986122886681096		
n	NPADE	ϵ-algorithm
2	1.0909090909090908e+00	1.0909090909090908e+00
3	1.0980392156862746e+00	1.0980392156862744e+00
4	1.0985703536493603e+00	1.0985703536493603e+00
5	1.0986092418124718e+00	1.0986092418124715e+00
6	1.0986120681169407e+00	1.0986120681169391e+00
7	1.0986122727383463e+00	1.0986122727383423e+00
8	1.0986122875191839e+00	1.0986122875191755e+00
9	1.0986122885853398e+00	1.0986122885853247e+00
10	1.0986122886621754e+00	1.0986122886621512e+00
11	1.0986122886677236e+00	1.0986122886676655e+00
12	1.0986122886681897e+00	1.0986122886680048e+00

Table 7.13: Test Case 7

Divergent Series, value = log(4) = 1.38629436111989061		
n	NPADE	ϵ-algorithm
2	1.3636363636363642e+00	1.3636363636363638e+00
3	1.3836734693877555e+00	1.3836734693877553e+00
4	1.3859968766267572e+00	1.3859968766267556e+00
5	1.3862608695652190e+00	1.3862608695652139e+00
6	1.3862906063093752e+00	1.3862906063093581e+00
7	1.3862939411876778e+00	1.3862939411876161e+00
8	1.3862943142284059e+00	1.3862943142281765e+00
9	1.3862943558893837e+00	1.3862943558885639e+00
10	1.3862943605376237e+00	1.3862943605357962e+00
11	1.3862943610560061e+00	1.3862943610555454e+00
12	1.3862943611037468e+00	1.3862943611203717e+00
13	1.3862943611100245e+00	1.3862943611326246e+00
14	1.3862943611376728e+00	1.3862943611501344e+00
15	1.3862943612165795e+00	1.3862943611021514e+00

Table 7.14: Test Case 8

Divergent Series, value = 1.939375420767		
n	NPADE	ϵ-algorithm
2	1.9142857142857144e+00	1.9142857142857141e+00
3	1.9365915430835006e+00	1.9365915430835001e+00
4	1.9390664468798502e+00	1.9390664468798493e+00
5	1.9393411129130236e+00	1.9393411129130227e+00
6	1.9393716102122101e+00	1.9393716102122085e+00
7	1.9393749974622545e+00	1.9393749974622558e+00
8	1.9393753737385362e+00	1.9393753737385611e+00
9	1.9393754155417327e+00	1.9393754155418290e+00
10	1.9393754201862419e+00	1.9393754201864437e+00
11	1.9393754207020171e+00	1.9393754207022478e+00
12	1.9393754207579967e+00	1.9393754207577061e+00
13	1.9393754207620360e+00	1.9393754207596183e+00
14	1.9393754207619309e+00	1.9393754207659415e+00
15	1.9393754207627061e+00	1.9393754207715170e+00

Table 7.15: Test Case 9

$\pi^2/6$, value = 1.644934066848226		
n	NPADE	ϵ-algorithm
2	1.5168539325842698e+00	1.5168539325842696e+00
3	1.5755129656131439e+00	1.5755129656131346e+00
4	1.6014586177636421e+00	1.6014586177635162e+00
5	1.6151702023410504e+00	1.6151702023395085e+00
6	1.6232861699299657e+00	1.6232861699144716e+00
7	1.6284839345148947e+00	1.6284839343240509e+00
8	1.6320117647709689e+00	1.6320117536780419e+00
9	1.6345153701315518e+00	1.6345149535410288e+00
10	1.6363559896029105e+00	1.6363456167441766e+00
11	1.6377477397687994e+00	1.6375865060640253e+00
12	1.6388057292192706e+00	1.6383204022120983e+00
13	1.6394002843582498e+00	1.6391132063748606e+00
14	1.6399916892474986e+00	1.6397130929514165e+00
15	1.6281807552595473e+00	1.6395813192524211e+00

Table 7.16: Test Case 10

Logarithmic Convergent Series, values = 0.75		
n	NPADE	ϵ-algorithm
2	5.6250000000000000e-01	5.6250000000000000e-01
3	6.5340909090909105e-01	6.5340909090908805e-01
4	6.9517241379313044e-01	6.9517241379276995e-01
5	6.9509107244958857e-01	6.9509107244931756e-01
6	7.1409910891606421e-01	7.1409910890279282e-01
7	7.2388908108986993e-01	7.2388908157043519e-01
8	7.1706098915603533e-01	7.1706130801666657e-01
9	7.3138365807020245e-01	7.3138370606731562e-01
10	7.3517771019368405e-01	7.3518218756934595e-01
11	7.3921311653686850e-01	7.4149383646776024e-01
12	7.3853192031585613e-01	7.3834325010621016e-01
13	7.3895919919500763e-01	7.3869790706926586e-01
14	7.4093680542023399e-01	7.3768254096611308e-01
15	7.4107140616006473e-01	7.3789761639108653e-01

Table 7.17: Test Case 11

Logarithmic Convergent Series, value = 2.223411647		
n	NPADE	ϵ-algorithm
2	2.0907439054969861e+00	2.0907439054969861e+00
3	2.1526678657013427e+00	2.1526678657013325e+00
4	2.1796052900733258e+00	2.1796052900731024e+00
5	2.1937561320973007e+00	2.1937561320963939e+00
6	2.2022196682748230e+00	2.2022196682570092e+00
7	2.2083018782056607e+00	2.2083018756312649e+00
8	2.2081629283753896e+00	2.2081629146261856e+00
9	2.2124218935954461e+00	2.2124217547634220e+00
10	2.2145358291121178e+00	2.2145311551069620e+00
11	2.2160389680509027e+00	2.2158534196782709e+00
12	2.2171846587229540e+00	2.2164036223543295e+00
13	2.2181603209356981e+00	2.2123793209125022e+00
14	2.2183482431533532e+00	2.2175348244225850e+00
15	2.2176619710327392e+00	2.2182446821198600e+00

Table 7.18: Test Case 12

Linear Monotone Sequence, value is 25.0		
n	NPADE	ϵ-algorithm
2	2.4999999999999769e+01	2.5000000000000234e+01
3	2.5000000000000082e+01	2.5000000000000234e+01
4	2.5000000000000060e+01	1.8999999999999790e+01
5	2.5000000000000028e+01	2.4999999999997840e+01

Table 7.19: Test Case 13

Linear Monotone Sequence; value = 20.0/3.0		
n	NPADE	ϵ-algorithm
2	6.6666666666666732e+00	6.6666666666667220e+00
3	6.6666666666666661e+00	6.6666666666666670e+00
4	6.6666666666666670e+00	6.6666666666666670e+00
5	6.6666666666666616e+00	6.6666666666666634e+00

Table 7.20: Test Case 14

log(5), value = 1.6094379124341003		
n	NPADE	ϵ-algorithm
2	1.5652173913043477e+00	1.5652173913043477e+00
3	1.6026936026936007e+00	1.6026936026936049e+00
4	1.6084303937881286e+00	1.6084303937881894e+00
5	1.6092887786830485e+00	1.6092887786832923e+00
6	1.6094159405464965e+00	1.6094159405466533e+00
7	1.6094346840307443e+00	1.6094346840286982e+00
8	1.6094374388826103e+00	1.6094374388492425e+00
9	1.6094378430572216e+00	1.6094378426083158e+00
10	1.6094379022998666e+00	1.6094378999721537e+00
11	1.6094379110298476e+00	1.6094379086370738e+00
12	1.6094379127046170e+00	1.6094379108362371e+00
13	1.6094379123291997e+00	1.6094379032466639e+00
14	1.6094379124036016e+00	1.6094379123095315e+00
15	1.6094379124332727e+00	1.6094379122493183e+00

Table 7.21: Test Case 15

Fourier Sine Series, value = 1.0707963267948966		
n	NPADE	ϵ-algorithm
2	8.6470402898219945e-01	8.6470402898219523e-01
3	1.1513569835463018e+00	1.1513569835463080e+00
4	1.0432069044671426e+00	1.0432069044671872e+00
5	1.0806664851217571e+00	1.0806664851221786e+00
6	1.0673267505376407e+00	1.0673267505388273e+00
7	1.0720219452392274e+00	1.0720219452392716e+00
8	1.0703644936619852e+00	1.0703644936449819e+00
9	1.0709485289206981e+00	1.0709485287903771e+00
10	1.0707427104767056e+00	1.0707427097292810e+00
11	1.0708152121552956e+00	1.0708152084582989e+00
12	1.0707896768060656e+00	1.0707896973336561e+00
13	1.0707986735142425e+00	1.0708016542775824e+00
14	1.0707955308239001e+00	1.0707973704121405e+00
15	1.0707968408997890e+00	1.0707976341291152e+00

Table 7.22: Test Case 16

$\frac{(1-z+z^3)}{(1-2z+z^2)}$, singularity at $z = 1.0$		
n	NPADE	ϵ-algorithm
2	-2.2204460492503131e-16	-2.2204460492503131e-16
3	-6.3246677575663212e+14	-2.0015998343868850e+15

Table 7.23: Test Case 17

ze^{2z} at $z = 1.0$, value = 7.3890560989306502		
n	NPADE	ϵ-algorithm
2	5.0000000000000009e+00	4.9999999999999964e+00
3	7.5000000000000009e+00	7.4999999999999858e+00
4	7.3870967741935436e+00	7.3870967741935472e+00
5	7.3890784982935145e+00	7.3890784982935083e+00
6	7.3890559230306661e+00	7.3890559230306714e+00
7	7.3890560999343693e+00	7.3890560999343826e+00
8	7.3890560989263028e+00	7.3890560989263125e+00
9	7.3890560989306620e+00	7.3890560989306584e+00
10	7.3890560989306460e+00	7.3890560989306584e+00

Table 7.24: Test Case 18

Perturbed Series Based on log(2)		
n	NPadé	ϵ-algorithm
2	6.9230769230769218e-01	6.9230769230769229e-01
3	6.9312169312169303e-01	6.9312169312169314e-01
4	6.9314641744548289e-01	6.9314641744548289e-01
5	Unstable node	Not a Number
6	Unstable node	Not a Number
7	Unstable node	Not a Number
8	Unstable node	Not a Number
9	6.9312638198117460e-01	Not a Number
10	6.9329151444854054e-01	Not a Number
11	6.9352183690707569e-01	Not a Number
12	6.9349564887372595e-01	Not a Number
13	6.9349585334925745e-01	Not a Number
14	6.9349578280244883e-01	Not a Number
15	6.9349578320935201e-01	Not a Number

Table 7.25: Test Case 19

Chapter 8

Summary and Suggestions for Further Research

The principal result of this thesis has been the development and analysis of an algorithm, NPADE, for computing Padé approximants that is weakly stable for the general class of power series. Other algorithms exist that are weakly stable for restricted classes of problems, but NPADE has no such restrictions. Moreover, except in exceptionally degenerate cases, NPADE is an $O(n^2)$ fast algorithm.

The relation that enabled so strong a conclusion to be drawn was (4.4), namely,

$$QU - VP = \vartheta(z) + r_0 v_0 z^{m+n-1},$$

where $\vartheta(z)$ depends on δR and δW, the errors in the order conditions. This

relation permitted, in Chapter 4, the derivation of the inverse of a Hankel matrix in terms of a Padé approximant and was also critical, in Chapter 5, in obtaining the bound on the propagation matrix $\mathcal{G}_i^{(k)}$. Due to this, the key value describing the condition number of the Hankel matrix and the size of $\mathcal{G}_i^{(k)}$ is $\frac{1}{|r_0 v_0|}$. The ε in the algorithm ensures that for any computed Padé approximant,

$$|r_0^{(k)} v_0^{(k)}| \geq \varepsilon.$$

Because of this, the error in the order condition, $\mathcal{D}^{(k)}$, and the error in the computed Padé approximants can be bounded by functions of ε. Moreover, $\mathcal{D}^{(k)}$ can be used to obtain an *a posteriori* estimate, (6.4), of the error in the computed Padé approximants.

An important question is how should ε be chosen? Equation (6.5) suggests that the loss of accuracy in the computed $(n-1, n)$ Padé approximant is approximately $\log_{10}(n\varepsilon^{-2})$ digits. This provides a rule-of-thumb for choosing ε, but it is based on an *a priori* estimate of the error and so may be somewhat pessimistic in the ε it suggests.

Lemma 4.4 shows how to obtain an approximate inverse to the Hankel matrix using the computed Padé approximants. If the system is well-conditioned, then NPADE can be used to obtain, in a stable fashion, the inverse of the Hankel matrix. An open question is whether this inverse formula can be extended to include the more general Sylvester matrix.

The analysis of Chapter 5 showed that the error in the final Padé approx-

imant, (U, V), is $O(\varepsilon^{-3})$. The experiments presented in Chapter 6, however, suggest that the error is $O(\varepsilon^{-2})$. Thus, one area for future research is the improvement of the bounds obtained in Chapter 5; in particular, it is suggested that the $O(\varepsilon^{-3})$ bound is an artifact of the analysis that is performed and not a feature of the actual problem.

As Wilkinson suggests [59], an *a priori* error analysis is useful to reveal where instabilities can arise and how they can be avoided. Our analysis accomplishes this, but it does not appear to describe completely what happens when instabilities are not avoided. In particular, the analysis suggests that when an unstable block is entered, the error in the order condition, $\mathcal{D}^{(k)}$, should grow; however, the experimental results of Chapter 6 suggest (e.g., $n_k = 23$ in Table 6.6) that the growth in the error does not occur until the block is left. It is conjectured that inside such blocks, the Sylvester residual systems are still well-conditioned and that the ill-conditioning does not appear until the very end of the block. Also, at the last point in an unstable block, the effect of the ill-conditioned Hankel system does not manifest itself (see $n_k = 22$ in Table 6.9). This may be related to the fact that at this point, the Padé approximant would still be unique even if the block were truly singular, as opposed to just being unstable. Additional study of the behavior of the algorithm within unstable blocks will likely reveal more about the computation of Padé approximants.

Another way of describing how NPADE operates, is to say that it uses an

analogue of Gaussian elimination with localized pivoting. In this regard, this makes NPADE a generalization of Rissanen's algorithm (which uses Gaussian elimination with no pivoting). It is an open question as to whether this analogy can be made precise and whether the techniques involved are applicable to any of the other algorithms cited in Chapter 2.

Chapter 7 showed the advantages of NPADE when used for sequence acceleration; namely, NPADE can produce results for unstable sequences. NPADE is equivalent to the ϵ-algorithm if the sequence being accelerated has no instabilities. An interesting question is whether or not the ideas that enable NPADE to perform stable computations can be applied to produce a stable variant of the ϵ-algorithm, or other algorithms.

In Chapter 7, the acceptance criteria for a solution was altered so that a solution was accepted if

$$\frac{|r_0^{(k+1)} v_0^{(k+1)}|}{< \hat{S}_k >} \geq \varepsilon,$$

where $\hat{S}_k$ is the Sylvester system solved to obtain $\hat{\mathcal{P}}^{(k)}$. This criteria functioned well because, in practice, the ratio of $|r_0^{(k+1)}|$ to $< \hat{S}_k >$ was, more or less, a constant if there were no instabilities. The problem when applying the analysis of Chapter 5 to convergent sequences is that the residuals, $\mathcal{R}^{(k)}$, tend to 0 as the algorithm computes Padé approximants of ever higher orders. A consequence of this is that while the Hankel system defined by the problem is becoming ever more ill-conditioned, the subproblems that are being solved to advance the approximants may still be well-conditioned. This aspect of

the algorithm is currently not reflected in the analysis, but it is believed that it is possible to capture this with further research. Such an investigation could also produce a different criteria for the selection of the stepsize, s_k. As the algorithm is stated, s_k is selected in an *a posteriori* manner. It may be possible to select s_k *a priori* directly from $\mathcal{R}^{(k)}$, so that Gaussian elimination would just have to be performed once for each iteration. This would make the algorithm exactly Gaussian elimination with partial pivoting in the totally unstable case. This was the initial direction of our research; it may prove to be fruitful now given the analysis of Chapters 4 and 5 as a starting point.

As remarked at the end of Chapter 3, NPADE is equivalent to the algorithm of Cabay and Choi [14] if exact arithmetic is used and ϵ is 0. In [14], Cabay and Choi give a superfast $O(n \log^2 n)$ version of their algorithm. It is an open question as to whether or not a larger stepsize and fast polynomial multiplication can be exploited by NPADE to produce a stable superfast variant. Also in [14], Cabay and Choi exhibit a duality between their algorithm and the Euclidean algorithm. An interesting question is what this duality may mean in a numerical setting.

One last question is whether or not the stability result can be strengthened. In the worst case, NPADE is numerically equivalent to Gaussian elimination with partial pivoting, which is a stable algorithm. We also know that as long as the final Sylvester system is nonsingular, NPADE produces a

solution (U, V) which has a small error in the order condition; viz,

$$AV - U = \delta W + z^{2n} W,$$

where $\| \delta W \|$ is small. These facts suggest that it may be possible to show that NPADE is actually stable, as opposed to simply weakly stable.

Appendix A

Details of Examples

A.1 Tests of Chapter 6

Test Case 1: This test is derived from a power series whose coefficients are uniformly distributed between -1 and $+1$. Perturbations of $O(10^{-1})$ were done to a_{16}, a_{20}, and a_{24} in order to enhance the instabilities occurring at the ninth, eleventh, and thirteenth nodes. This series is presented in Table A.26.

Test Case 2: This test is identical to Test Case 1, except that a_{38} was perturbed by $O(10^{-1})$ in order to make the instability at the twentieth node more pronounced. This series is presented in Table A.27.

Test Case 3: This test is derived from Test Case 1. In this case a perturbation of $O(10^{-1})$ is made to a_{20} to remove the instability at the eleventh

node. Then $O(1)$ perturbations are done to a_{36}, a_{37}, a_{38}, and a_{39} in order to increase the instability at the twentieth node. This series is given in Table A.28.

A.2 Sequence Acceleration Test Cases

We have selected a number of test problems from the literature. Most are drawn from Smith and Ford [52], [53] as their test cases are based upon a survey of the literature. They classified their test cases into a number of groupings; the experiments presented use a cross section of their examples, plus a handful of sequences from other sources.

Test Case 4: Smith and Ford [52]; alternating series.

$$\log(2) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{1}{n} = 0.693147180559945\ldots$$

Test Case 5: Leibnitz series; Shanks [51]; alternating series.

$$\pi = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{4}{2n-1} = 3.1415926535898\ldots$$

Test Case 6: Levin [40]; alternating series.

$$(1-\sqrt{2})\zeta(1/2) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{1}{\sqrt{n}} = 0.60489864342162\ldots$$

Test Case 7: Levin [40], Smith and Ford [53]; divergent series.

$$\log(3) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{2^n}{n} = 1.0986122886681096\ldots$$

Test Case 8: Smith and Ford [53]; divergent series.

$$\log(4) = \sum_{n=1}^{\infty}(-1)^{n+1}\frac{3^n}{n} = 1.38629436111989061\ldots$$

Test Case 9: Smith and Ford [53]; divergent series.

$$\int_1^4 \frac{\log(t)}{t-1}dt = \sum_{n=1}^{\infty}(-1)^{n+1}\frac{3^n}{n^2} = 1.939375420767\ldots$$

Test Case 10: Levin [40], Smith and Ford [52]; logarithmic convergence.

$$\frac{\pi^2}{6} = \sum_{n=1}^{\infty}\frac{1}{n^2} = 1.644934066848226\ldots$$

Test Case 11: Smith and Ford [52]; logarithmic convergence.

$$\frac{3}{4} = \sum_{n=1}^{\infty}\frac{2n-1}{n(n+1)(n+2)}$$

Test Case 12: Smith and Ford [52]; logarithmic convergence.

$$\sum_{n=1}^{\infty}\frac{1+n^2+n^4}{n^2(1+n^4)} = 2.223411647\ldots$$

Test Case 13: Smith and Ford [52]; linear monotone convergent sequence.

$$25 = \sum_{n=1}^{\infty} n(.8)^{n-1}$$

Test Case 14: Smith and Ford [52]; linear monotone convergent sequence.

$$\frac{20}{3} = \sum_{n=1}^{\infty} .4^{n-1} + .8^{n-1}$$

Test Case 15: Smith and Ford [52]; linear monotone convergent sequence.

$$\log(5) = \sum_{n=1}^{\infty}\frac{.8^n}{n} = 1.6094379124341003$$

Test Case 16: Smith and Ford [53]; Fourier sine series.

$$\frac{(\pi - 1)}{2} = \sum_{n=1}^{\infty} \frac{\sin(n)}{n} = 1.070796326794896\ldots$$

Test Case 17: Gragg [28]; rational function, singularity at $z = 1.0$.

$$\frac{(1 - z + z^3)}{(1 - 2z + z^2)} = \infty$$

Test Case 18: Wynn [64]; exponential function.

$$e^2 = \sum_{n=0}^{\infty} \frac{2^n}{n!} = 7.3890560989306502\ldots$$

Test Case 19: This test case was derived from Test Case 4 (log(2)) by perturbing four coefficients in order to introduce an instability into the sequence. The perturbations were $O(10^{-2})$ and the original and perturbed values are given in Table A.29.

n	a_n	n	a_n
0	1.700998006110077100e-02	37	-8.750988811953914800e-01
1	-7.306280664453543800e-01	38	-4.099112147557653100e-01
2	-2.415133549287107300e-02	39	8.141043137890327000e-01
3	9.361102999524306700e-02	40	-5.779342070016397800e-01
4	-4.240208691209749200e-01	41	-4.670527879773199700e-01
5	1.349156835441622800e-01	42	-3.004388988953445900e-01
6	4.142049266157716800e-01	43	1.590260771652927100e-01
7	-3.787970304291667900e-01	44	-6.867059969219435600e-01
8	-4.777310206347433800e-01	45	-7.582574531047208100e-01
9	-8.165882237410062600e-01	46	6.156336903717682900e-01
10	2.584255954794834100e-02	47	-9.995655557136662300e-01
11	-6.232951084368816300e-01	48	8.926375134779489500e-01
12	-6.811659538011680700e-01	49	4.367033470763857000e-01
13	-7.222180708518419800e-01	50	-4.352120118543617600e-01
14	3.150292572705346600e-01	51	-4.688462190971208800e-01
15	6.675804431248273400e-01	52	-8.801535050199865700e-01
16	6.887006678867160200e-01	53	-8.528973579899383800e-01
17	-9.765136921559588500e-01	54	8.867103922057062400e-01
18	-2.694635049155573700e-01	55	-2.400888411701552800e-01
19	-3.962317578459454400e-01	56	8.337901517132207 00e-01
20	3.608431656126335200e-01	57	2.204355152514163600e-01
21	5.929501732745667800e-02	58	3.651027952927190500e-01
22	4.847313328503923800e-01	59	-8.053140293856282400e-01
23	3.687748688913750100e-03	60	-6.394915633271369700e-01
24	8.396350667994796300e-01	61	4.491280731271283200e-01
25	5.629033302542785200e-01	62	1.639618958942237200e-01
26	7.906369546341123200e-01	63	5.221532904749302000e-01
27	8.588388626080367700e-01	64	7.016496134005949500e-01
28	-2.394137254351880200e-02	65	-7.129098109089860900e-01
29	-5.911093247971583900e-01	66	-9.431727639801547000e-01
30	2.044619705569577000e-01	67	2.207209749340275400e-01
31	2.944952866942578400e-01	68	-5.235962099615449900e-01
32	-8.826192858467057900e-01	69	-4.063880112081654800e-01
33	4.165233750050174600e-01	70	-3.635019952091407300e-01
34	7.610984647284246800e-01	71	4.466053288864024800e-02
35	9.639916410334330400e-01	72	-1.811777066264094200e-01
36	-8.547807958487241300e-02	73	-1.578959228078796700e-01

Table A.26: Series 1

n	a_n	n	a_n
0	1.70099800611007710e-02	37	-8.75098881195391480e-01
1	-7.30628066445354380e-01	38	-6.49627650313868820e-01
2	-2.41513354928710730e-02	39	8.14104313789032700e-01
3	9.36110299952430670e-02	40	-5.77934207001639780e-01
4	-4.24020869120974920e-01	41	-4.67052787977319970e-01
5	1.34915683544162280e-01	42	-3.00438898895344590e-01
6	4.14204926615771680e-01	43	1.59026077165292710e-01
7	-3.78797030429166790e-01	44	-6.86705996921943560e-01
8	-4.77731020634743380e-01	45	-7.58257453104720810e-01
9	-8.16588223741006260e-01	46	6.15633690371768290e-01
10	2.58425595479483410e-02	47	-9.99565555713666230e-01
11	-6.23295108436881630e-01	48	8.92637513477948950e-01
12	-6.81165953801168070e-01	49	4.36703347076385700e-01
13	-7.22218070851841980e-01	50	-4.35212011854361760e-01
14	3.15029257270534660e-01	51	-4.68846219097120880e-01
15	6.67580443124827340e-01	52	-8.80153505019986570e-01
16	6.88700667886716020e-01	53	-8.52897357989938300e-01
17	-9.76513692155958850e-01	54	8.86710392205706240e-01
18	-2.69463504915557370e-01	55	-2.40088841170155280e-01
19	-3.96231757845945440e-01	56	8.33379015171322070e-01
20	3.60843165612633520e-01	57	2.20435515251416360e-01
21	5.92950173274566780e-02	58	3.65102795292719050e-01
22	4.84731332850392380e-01	59	-8.05314029385628240e-01
23	3.68774868891375010e-03	60	-6.39491563327136970e-01
24	8.39635066799479630e-01	61	4.49128073127128320e-01
25	5.62903330254278520e-01	62	1.63961895894223720e-01
26	7.90636954634112320e-01	63	5.22153290474930200e-01
27	8.58838862608036770e-01	64	7.01649613400594950e-01
28	-2.39413725435188020e-02	65	-7.12909810908986090e-01
29	-5.91109324797158390e-01	66	-9.43172763980154700e-01
30	2.04461970556957700e-01	67	2.20720974934027540e-01
31	2.94495286694257840e-01	68	-5.23596209961544990e-01
32	-8.82619285846705790e-01	69	-4.06388011208165480e-01
33	4.16523375005017460e-01	70	-3.63501995209140730e-01
34	7.61098464728424680e-01	71	4.46605328886402480e-02
35	9.63991641033433040e-01	72	-1.81177706626409420e-01
36	-8.54780795848724130e-02	73	-1.57895922807879670e-01

Table A.27: Series 2

n	a_n	n	a_n
0	1.70099800611007712 1e-02	37	8.458446387224216600e-02
1	-7.30628066445354273 9e-01	38	-1.280862014690811179e-01
2	-2.41513354928710732 8e-02	39	-5.621072487234184401e-01
3	9.36110299952430668 9e-02	40	-5.779342070016397770e-01
4	-4.24020869120974919 2e-01	41	-4.670527879773199720e-01
5	1.34915683544162279 3e-01	42	-3.004388988953445860e-01
6	4.14204926615771684 7e-01	43	1.590260771652927119e-01
7	-3.78797030429166736 7e-01	44	-6.867059969219435622e-01
8	-4.77731020634743375 2e-01	45	-7.582574531047208133e-01
9	-8.16588223741006258 3e-01	46	6.156336903717682940e-01
10	2.58425595479483405 4e-02	47	-9.995655557136662317e-01
11	-6.23295108436881628 6e-01	48	8.926375134779489517e-01
12	-6.81165953801168067 0e-01	49	4.367033470763856950e-01
13	-7.22218070851941975 6e-01	50	-4.352120118543617577e-01
14	3.15029257270534657 2e-01	51	-4.688462190971209309e-01
15	6.67580443124827338 5e-01	52	-8.801535050199866816e-01
16	6.88700667886716022 5e-01	53	-8.528973579899384871e-01
17	-9.76513692155958845 7e-01	54	8.867103922057063548e-01
18	-2.69463504915557372 5e-01	55	-2.400888411701552760e-01
19	-3.96231757845945442 6e-01	56	8.333790151713220684e-01
20	-7.83059656346086097 0e-01	57	2.204355152514163585e-01
21	5.92950173274566713 6e-02	58	3.651027952927190512e-01
22	4.84731332850392382 4e-01	59	-8.053140293856282383e-01
23	3.68774868891375050 6e-03	60	-6.394915633271369693e-01
24	8.39635066799479745 6e-01	61	4.491280731271283799e-01
25	5.62903330254278522 2e-01	62	1.639618958942237192e-01
26	7.90636954634112321 1e-01	63	5.221532904749301984e-01
27	8.58838862608036879 3e-01	64	7.016496134005949514e-01
28	-2.39413725435188053 1e-02	65	-7.129098109089860857e-01
29	-5.91109324797158386 4e-01	66	-9.431727639801547047e-01
30	2.04461970556957722 3e-01	67	2.207209749340275362e-01
31	2.94495286694257840 7e-01	68	-5.235962099615449894e-01
32	-8.82619285846705792 2e-01	69	-4.063880112081654783e-01
33	4.16523375005017404 4e-01	70	-3.635019952091407269e-01
34	7.61098464728424573 0e-01	71	4.466053288864024751e-02
35	9.63991641033433044 3e-01	72	-1.811777066264094183e-01
36	2.84526725661764112 7e+00	73	-1.578959228078796739e-01

Table A.28: Series 3

n	Old Value	New Value
9	1.11111111111111000e-01	1.110884353741496827e-01
10	-1.00000000000000100e-01	-9.989795918367352923e-02
11	9.090909090909091200e-02	9.064139941690971258e-02
12	-8.333333333333332900e-02	-8.279640427599622987e-02

Table A.29: Details of the Perturbation of Test Case 4

Bibliography

[1] Gregory S. Ammar and William B. Gragg. Superfast solution of real positive definite Toeplitz systems. *SIAM Journal on Matrix Analysis and Its Applications*, 9:61–76, 1988.

[2] G. Baker and P. Graves-Morris. *Padé Approximants*, volume 14 of *Encyclopedia of Mathematics*. Addison-Wesley, 1981.

[3] G. Baker and P. Graves-Morris. *Padé Approximants*, volume 13 of *Encyclopedia of Mathematics*. Addison-Wesley, 1981.

[4] Erwin H. Bareiss. Numerical solution of linear equations with Toeplitz and vector matrices. *Numerische Mathematik*, 13:404–424, 1969.

[5] Robert R. Bitmead and Brian D. O. Anderson. Asymptotically fast solutions of Toeplitz and related systems of linear equations. *Linear Algebra and Its Applications*, 34:103–116, 1980.

[6] R. Brent, F. Gustavson, and D. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *Journal of Algorithms*, 1:259–295, 1980.

[7] C. Brezinski. Computation of Padé approximants and continued fractions. *Journal of Computational and Applied Mathematics*, 2:113–123, 1976.

[8] C. Brezinski and A. C. Rieu. The solution of systems of equations using the ϵ-algorithm, and an application to boundary-value problems. *Mathematics of Computation*, 28:731–741, 1974.

[9] A. Bultheel. Recursive algorithms for nonnormal Padé tables. *SIAM Journal on Applied Mathematics*, 39:106–118, 1980.

[10] A. Bultheel and L. Wuytack. Stability of numerical methods for computing Padé approximants. *in Approximation Theory III*, pages 267–272, 1980.

[11] James R. Bunch. Stability of methods for solving Toeplitz systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 6:349–364, 1985.

[12] James R. Bunch. The weak and strong stability of algorithms in numerical linear algebra. *Linear Algebra and Its Applications*, 88/89:49–66, 1987.

[13] Stan Cabay and T. T. C. Kao. The diagonal Padé table and the triangular decomposition of hankel matrices. Technical Report TR83-10, Dept. Comp. Sci. Univ. Alberta, 1983.

[14] Stanley Cabay and Dong-Koo Choi. Algebraic computations of scaled Padé fractions. *SIAM Journal on Computing*, 15:243–270, 1986.

[15] G. Claessens. A new look at the Padé table and the different methods for computing its elements. *Journal of Computational and Applied Mathematics*, 1:141–153, 1975.

[16] G. Claessens and L. Wuytack. On the computation of non-normal Padé approximants. *Journal of Computational and Applied Mathematics*, 5:282–289, 1979.

[17] F. Cordellier. Demonstration algébrique de l'extesion de l'identité de wynn aux tables de Padé nonnormales. In *Padé Approximation and its Applications*, pages 36–60. Springer-Verlag, 1979.

[18] G. Cybenko. The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 1:303–319, 1980.

[19] Frank de Hoog. A new algorithm for solving Toeplitz systems of equations. *Linear Algebra and Its Applications*, 88/89:123–138, 1987.

[20] Philippe Delsarte and Yves V. Genin. The split Levinson algorithm. *IEEE Trans. ASSP*, 34:470–478, 1986.

[21] Michael E. Fisher. Critical phenomena - series expansions and their analysis. In *Padé Approxmimants and Their Application*, pages 159–162. Academic Press, 1973.

[22] George E. Forsythe and Cleve B. Moler. *Computer Solution of Linear Algebraic Systems.* Prentice-Hall, 1967.

[23] G. Frobenius. Relationen zwischen den näherungsbrüchen von potenzreihen. *J. für Math.*, 30:1–17, 1881.

[24] K. O. Geddes. Symbolic computation of Padé approximants. *ACM Transactions on Mathematical Software*, 5:218–233, 1979.

[25] E. Gekeler. On the solution of systems of equations by the epsilon algorithm of wynn. *Mathematics of Computation*, 26:427–436, 1972.

[26] I. C. Gohberg and A. A. Semencul. On the inversion of finite Toeplitz matrices and their continuous analogs. *Mat. Issled.*, 2:201–233, 1972.

[27] Gene H. Golub and Charles F. van Loan. *Matrix Computations.* Johns Hopkins University Press, 1983.

[28] William B. Gragg. The Padé table and its relation to certain algorithms of numerical analysis. *SIAM Review*, 14:1–62, 1972.

[29] P. Graves-Morris. The numerical calculation of Padé approximants. In *Padé Approximation and its Applications*, pages 231–245. Springer-Verlag, 1979.

[30] G. H. Hardy. *Divergent Series.* Oxford University Press, 1956.

[31] Nicholas J. Higham and Desmond J. Higham. Large growth factors in gaussian elimination with pivoting. *SIAM Journal on Matrix Analysis and Its Applications*, 10:155–164, 1989.

[32] I.S. Iohvidov. *Hankel and Toeplitz Matrices and Forms - Algebraic Theory.* Birkhauser, 1982.

[33] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods.* Wiley, 1966.

[34] Jaswant R. Jain. An efficient algorithm for a large Toeplitz set of linear equations. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, AASP-27:612–615, 1979.

[35] D. Kahaner. Numerical quadrature by the ϵ-algorithm. *Mathematics of Computation*, 26:689–693, 1972.

[36] Thomas Kailath. A view of three decades of linear filtering theory. *IEEE Trans. Info. Th.*, 20:146–181, 1974.

[37] Piotr Kossowski. *Modular Padé Forms for Bivariate Power Series.* PhD thesis, Dept. Comp. Sci., Univ. Alberta, 1986.

[38] Rajendra Kumar. A fast algorithm for solving a Toeplitz system of equations. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-33:254–267, 1985.

[39] George Labahn, Dong Koo Choi, and Stan Cabay. The inverses of block hankel and block Toeplitz matrices. *SIAM Journal on Computing*, 19:98–123, 1990.

[40] D. Levin. Development of non-linear transformations for improving convergence of sequences. *Intl. J. Comp. Math.*, 3:371–388, 1973.

[41] N. Levinson. The Weiner RMS error criterion in filter design and prediction. *Stud. Appl. Math.*, 25:261–278, 1947.

[42] Franklin T. Luk and Sanzheng Qiao. A fast but unstable orthogonal triangularization technique for Toeplitz matrices. *Linear Algebra and Its Applications*, 88/89:495–506, 1987.

[43] Y. L. Luke. Computations of coefficients in the polynomials of Padé approximations by solving systems of linear equations. *Journal of Computational and Applied Mathematics*, 6:213–218, 1980.

[44] Y. L. Luke. A note on evaluation of coefficients in the polynomials of Padé approximants by solving systems of linear equations. *Journal of Computational and Applied Mathematics*, 8:93–99, 1982.

[45] John Makhoul. Linear prediction: A tutorial review. *Proc. IEEE*, 63:561–580, 1975.

[46] H. Padé. Sur la représentation approchés d'une fonction par des fractions rationelles. *Ann. Ec. Norm.*, Sup. 3:1–93, 1892.

[47] M. Pindor. Padé approximants and rational functions as tools for finding poles and zeros of analytical functions measured experimentally. In *Padé Approximation and its Applications*, pages 338–351. Springer-Verlag, 1979.

[48] M. F. Reusch, L. Ratzan, N. Pomphrey, and W. Park. Diagonal Padé approximations for initial value problems. *SIAM Journal on Scientific and Statistical Computing*, 9:829–838, 1988.

[49] Jorma Rissanen. Solution of linear equations with hankel and Toeplitz matrices. *Numerische Mathematik*, 22:361–366, 1974.

[50] Y. Shamesh. Approximation of linear time-invariant systems. In *Padé Approximants and Their Applications*, pages 341–345. Academic Press, 1973.

[51] D. Shanks. Non-linear transformations of divergent and slowly convergent sequences. *J. Math. and Phys.*, 34:1–42, 1955.

[52] D. A. Smith and W. F. Ford. Acceleration of linear and logarithmic convergence. *SIAM Journal on Numerical Analysis*, 16:223–240, 1979.

[53] D. A. Smith and W. F. Ford. Numerical comparison of nonlinear convergence accelerators. *Mathematics of Computation*, 38:481–499, 1982.

[54] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, 1973.

[55] Douglas R. Sweet. Fast Toeplitz orthogonalization. *Numerische Mathematik*, 43:1–21, 1984.

[56] L. N. Trefethen and R. S. Schreiber. Average-case stability of Gaussian elimination. *SIAM Journal on Matrix Analysis and Its Applications*, 11:to appear, 1990.

[57] W. Trench. An algorithm for the inversion of finite hankel matrices. *SIAM*, 13:1102–1107, 1965.

[58] G. Walker. On periodicity in series of related terms. *Proc. Roy. Soc. London, Ser. A*, 131A:518, 1931.

[59] James H. Wilkinson. Modern error analysis. *SIAM Review*, 13:548–568, 1971.

[60] J.H. Wilkinson. *Rounding Errors in Algebraic Processes.* Prentice-Hall, 1963.

[61] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[62] Jet Wimp. *Sequence Transformations and their Applications.* Academic Press, 1981.

[63] P. Wynn. On a device for computing the $e_m(s_n)$ transformation. *Math. Tables Aides Comput.*, 10:91–96, 1956.

[64] P. Wynn. Singular rules for certain non-linear algorithms. *BIT*, 3:175–195, 1963.

[65] P. Wynn. On the convergence and stability of the epsilon algorithm. *SIAM Journal on Numerical Analysis*, 3:91–122, 1966.

[66] P. Wynn. Upon systems of recursions which obtain among the quotients of the Padé table. *Numerische Mathematik*, 8:264–269, 1966.

[67] G. U. Yule. On a method of investigating periodicities in disturbed series, with special reference to wolfe's sunspot numbers. *Philos. Trans. Roy. Soc. London, Ser. A.*, 226A:267–298, 1927.

[68] S. Zohar. Toeplitz matrix inversion: The algorithm of w.f. trench. *JACM*, 16:592–601, 1969.

[69] S. Zohar. The solution of a Toeplitz set of linear equations. *JACM*, 21:272–276, 1974.